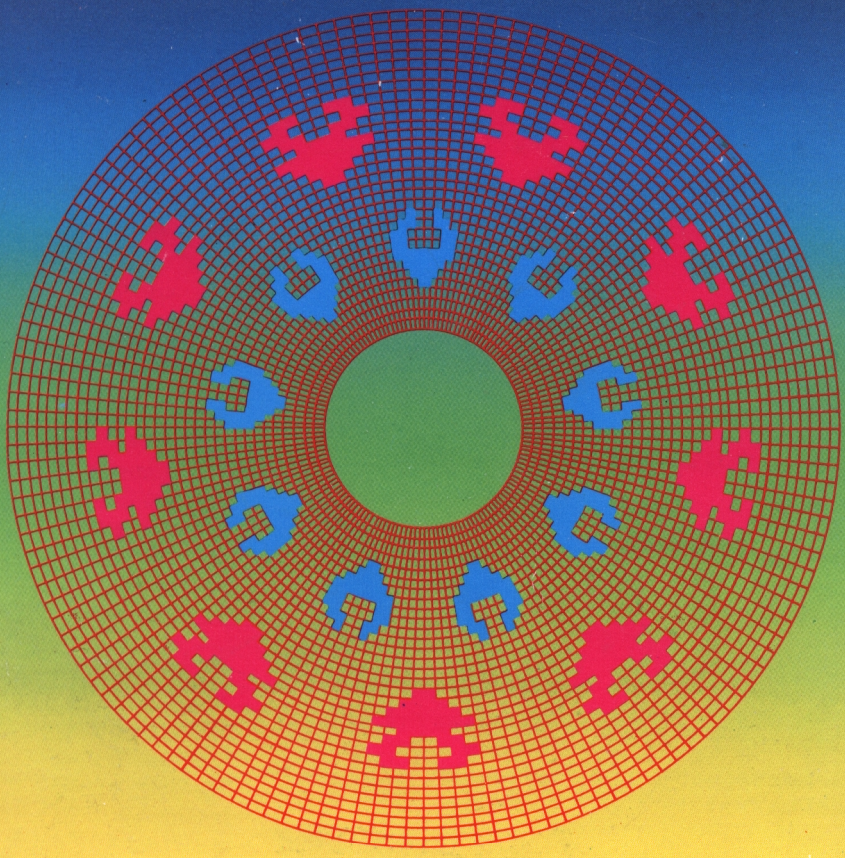
# THE ORIC BOOK OF GAMES



MIKE JAMES,
S.M.GEE AND KAY EWBANK

**Other books of interest from Granada:**

## The Apple II

**APPLE II PROGRAMMERS HANDBOOK**
R. C. Vile
0 246 12027 4

## ATARI

**GET MORE FROM THE ATARI**
Ian Sinclair
0 246 12149 1

**THE ATARI BOOK OF GAMES**
M. James, S. M. Gee and K. Ewbank
0 246 12277 3

## The BBC Micro

**INTRODUCING THE BBC MICRO**
Ian Sinclair
0 246 12146 7

**THE BBC MICRO – AN EXPERT GUIDE**
Mike James
0 246 12014 2

**21 GAMES FOR THE BBC MICRO**
M. James, S. M. Gee and K. Ewbank
0 246 12103 3

**BBC MICRO GRAPHICS AND SOUND**
Steve Money
0 246 12156 4

**DISCOVERING BBC MICRO MACHINE CODE**
A. P. Stephenson
0 246 12160 2

**THE BBC BASIC PROGRAMMER**
S. M. Gee and M. James
0 246 12158 0

**BBC MICRO AND ELECTRON MACHINE CODE – AN EXPERT GUIDE**
A. P. Stephenson and D. J. Stephenson
0 246 12227 7

## The Colour Genie

**MASTERING THE COLOUR GENIE**
Ian Sinclair
0 246 12190 4

## The Commodore 64

**COMMODORE 64 COMPUTING**
Ian Sinclair
0 246 12030 4

**THE COMMODORE 64 GAMES BOOK**
Owen Bishop
0 246 12258 7

**SOFTWARE 64 Practical Programs for the Commodore 64**
Owen Bishop
0 246 12266 8

## The Dragon 32

**THE DRAGON 32 And How to Make The Most Of It**
Ian Sinclair
0 246 12114 9

**THE DRAGON 32 BOOK OF GAMES**
M. James, S. M. Gee and K. Ewbank
0 246 12102 5

**THE DRAGON PROGRAMMER**
S. M. Gee
0 246 12133 5

**DRAGON GRAPHICS AND SOUND**
Steve Money
0 246 12147 5

**THE DRAGON 32 How to Use and Program**
Ian Sinclair
0 586 06103 7

## The Electron

**THE ELECTRON PROGRAMMER**
S. M. Gee and Mike James
0 246 12340 0

**21 GAMES FOR THE ELECTRON**
Mike James, S. M. Gee and K. Ewbank
0 246 12344 3

**BBC MICRO AND ELECTRON MACHINE CODE – AN EXPERT GUIDE**
A. P. Stephenson and D. J. Stephenson
0 246 12227 7

**Learning is Fun! 40 EDUCATIONAL GAMES FOR THE BBC MICRO AND ELECTRON**
Vince Apps
0 246 12317 6

## The IBM Personal Computer

**THE IBM PERSONAL COMPUTER**
James Aitken
0 246 12151 3

## The Jupiter Ace

**THE JUPITER ACE**
Owen Bishop
0 246 12197 1

## The Lynx

**LYNX COMPUTING**
Ian Sinclair
0 246 12131 9

## The NewBrain

**THE NEWBRAIN And How To Make The Most Of It**
Francis Samish
0 246 12232 3

## The ORIC-1

**THE ORIC-1 And How To Get The Most From It**
Ian Sinclair
0 246 12130 0

**THE ORIC-1 BOOK OF GAMES**
M. James, S. M. Gee and K. Ewbank
0 246 12155 6

Continued on inside back cover

# The Oric Book of Games

# The Oric Book of Games

M. James, S. M. Gee and K. Ewbank

GRANADA
London Toronto Sydney New York

Granada Technical Books
Granada Publishing Ltd
8 Grafton Street, London W1X 3LA

First published in Great Britain by
Granada Publishing 1984



ISBN 0-246-12155-6

Typeset by V & M Graphics Ltd, Aylesbury, Bucks
Printed and bound in Great Britain
by Mackays of Chatham, Kent

# Contents

# Introduction

This is a collection of twenty-one games written to be played on your 16K or 48K Oric. Every game is complete in itself so you can turn to whichever one takes your fancy, type it in and play it. We've tried to include something for everyone and each one has its own detailed description so that you'll know what to expect before you embark on it. You also have a chance to *see* what to expect as there are samples of the displays produced on your TV screen. Of course these cannot really do justice to many of the programs which use colour graphics – and we cannot find any way of letting you hear the accompanying sound effects.

All the games are written in BASIC and, presented in this form, they serve a dual purpose. Typing in games for yourself is a good way to *absorb* BASIC. If you are a beginner you will soon become familiar with its syntax and structure and if you already have some experience you will quickly pick up some handy techniques that you can incorporate into your own programs.

If you are not very experienced in using your Oric, try typing in the shorter programs first. Our only advice is to save your programs often and make more than one copy when you do so – we normally save each version three times – including a copy at the slower speed. Save the version you expect to be the final one *before* you attempt to RUN the program. Also, do not let your Oric overheat. If it gets too warm, save your program and go for a break yourself, switching your Oric off to let it have a rest too. Never leave your machine switched on unused for a long period.

### What's to follow

It's really impossible to indicate the range of programs included in this book as they do not fall into neat categories. Of the twenty-one

programs about two-thirds can be described as moving graphics games. Some of these are variations on familiar favourites, for example Alien Invaders, Rainbow Squash and Bobsleigh. Others have titles that probably won't ring any bells – Sheepdog Trials, Commando Jump and Across the Ravine – but we hope they will soon become popular once you start to play them. Laser Attack and Mighty Missile are both 'zap-the-enemy' type games with special features that make them very different from others we've played. Treasure Island is another program that is out of the ordinary. It is a game that tests your memory and relies on a variety of interesting graphics techniques. Oric Ledger presents a horse race run to a familiar and appropriate tune and Capture the Quark is a board game in which you play against the computer on an eight-by-eight grid. There are also some programs for traditional pastimes – Magic Dice, Noughts and Crosses, Word Scramble and Mirror Tile all come into this category. Oric Oracle is a rather unusual program that enables your Oric to hold a *conversation* with you. If you think we are joking you'll have to try it for yourself.

All the programs in this book run on either a 48K or a 16K Oric. However, when you try to enter 'Oric Oracle' on a 16K Oric you will find you run out of memory. There is a simple remedy for this which involves using the memory normally set aside for the high resolution graphics screen and details of this are given in the program description.

## Improve your programming

This book isn't however intended as just another collection of programs. As well as providing programs that you can have hours of fun with we also hope to cater for all Oric owners by presenting a book that can be used in more than one way. True, you can use it simply as a source of exciting games programs, but on the other hand you can use it to further improve your own knowledge of BASIC programming. Each program is accompanied by an outline of its subroutine structure, details of special programming techniques and suggestions for further improvements. These sections are included for those of you who want to develop your own programming skills. By giving away some of our *trade secrets* we hope that you'll be able to extend your range of techniques.

It is because we would like to be able to help you experiment with your own programming that all these games stick to BASIC. This

means, of course, that the games cannot be as fast-moving or as complicated as the ones that are available pre-programmed on cassettes which are written in machine code. But if you want to learn to write your own programs it is far easier to start with BASIC than to attempt to come to terms with machine code.

## Perfect programming

The programs included have all been extensively tested in the form in which they appear and then printed out directly from working versions. This means that if you type in exactly what is printed in this book every program should work for you every time you run it. You will no doubt quickly notice that the programs in this book are not neatly numbered in even steps. The reason for the uneven line numbering employed is that the Oric does not have a line renumbering facility and although it is tempting to neaten up the line numbers as you type the programs back in we strongly advise against it. All sorts of errors can creep in as a result and it will be much more difficult to trace them if you deviate from the listings.

However careful you are, it's almost impossible to avoid introducing bugs when you re-enter a program – so if a program won't work when you've typed in it check it against the listing and if it still won't work have a cup of tea and check again – it's all too easy to read what you think should be there rather than what is there! If there are any particular points to look out for when entering a program you'll be alerted to them in the section on Typing Tips. Common sources of possible errors are, confusing full stops and commas or semi-colons and colons, omitting brackets (or putting in extra ones) or mis-reading less than and greater than symbols – getting these round the wrong way will lead to chaotic results.

If you do get an error message when you try to RUN a program, don't just give up but use the information it gives you to trace your mistake. The error will not necessarily be in the line whose number is reported but that line will try to use some part of the program with the bug in it. If the line uses a variable or an array check to see if it was defined properly. If the line identified goes to another part of the program or calls a procedure or a subroutine make sure that section is complete.

## Cassette tapes

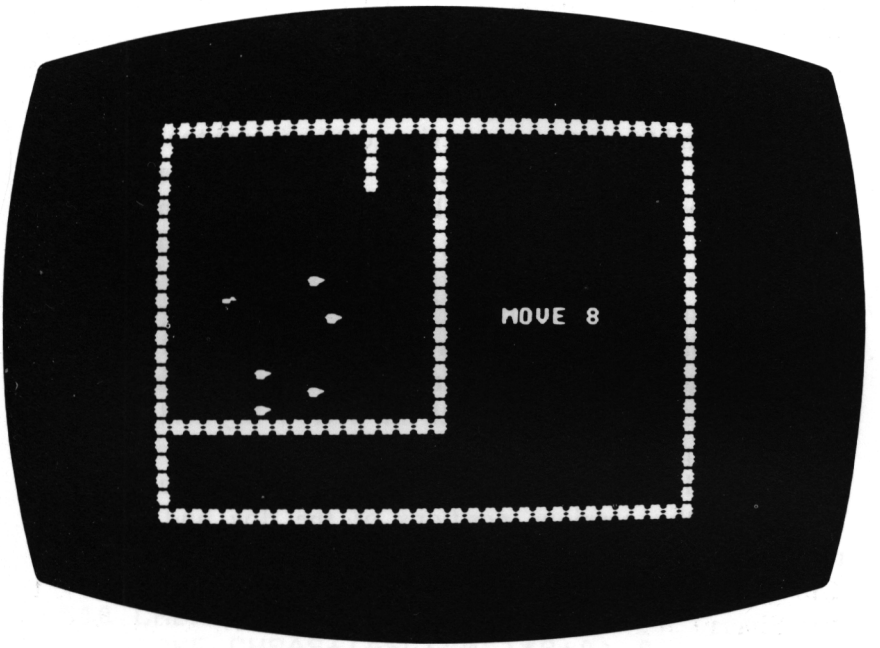
Typing in long programs can be a very frustrating business. It's not easy to avoid typing mistakes altogether and there is always the risk that you'll lose your program before you've saved it after hours of careful effort – it's far too easy to disconnect your power supply and you can't guard against thunderstorms or other sources of voltage fluctuations without great expense! Many of the programs in this book are indeed long. This is unavoidable as the games concerned include lots of features. But you can avoid having to type them all in for yourselves. The programs, exactly as listed here, are available on a pair of cassette tapes. For full details and an order form send a stamped, self-addressed envelope to

RAMSOFT,
P.O. Box 6,
Richmond,
North Yorkshire,
DL10 4HL.

## Acknowledgements

Our grateful thanks are due to Oric Products International and to Ian Sinclair for loaning us equipment on which to develop the programs in this book and to Richard Miles and Julian Grover of Granada for their help throughout the project.

# 1
# Sheepdog Trials




If you've ever watched a shepherd and his dog coax a flock of sheep into a pen you're bound to agree that its a quite astounding feat. The experienced shepherd makes it all look so effortless as he shouts and whistles commands to his dog who, obediently stands his ground, or edges up a few paces, or runs around the back of the flock to head off a straggler.

This game is an extremely realistic simulation. In fact it's so true-to-life that the only person we know who's scored a "Super Shepherd" rating was a real sheep farmer!! Five fluffy white sheep and a dog appear in a green field surrounded by a picket fence – it creates just the right country atmosphere.

## How to play

The object of the game is to herd all five sheep into the pen at the top

right-hand side of their field in the minimum number of moves. To do this you have to control the dog by using the arrow keys. If the dog approaches too close to the sheep they will scatter (they may also scatter randomly during the course of the game just to complicate matters). In normal play neither the dog nor the sheep are allowed to cross any fences, although when they scatter the sheep may jump out of the pen. There will always be a total of five sheep but if they crowd very close together they will appear to merge into one another.

Once you've played this game a few times you'll realise that some strategies for controlling the sheep work better than others. Beginners tend to waste moves trying to manoeuvre the dog around the back of the flock. However, to achieve the title of "Super Shepherd" or "Good dog" you'll need to make every move count.

## Typing tips

The hash character is used to print the fence in subroutine 800. It is produced by typing the 3 key with the 'SHIFT' key held down. The only other printing feature to look out for is the single space enclosed in double quotes in lines 1600 and 7180. These are used to blank out the previous positions of the dog and the sheep respectively.

## Subroutine structure

| | |
|---|---|
| 500 | Sets up graphics characters and arrays |
| 800 | Prints fences |
| 1000 | Prints sheep |
| 1100 | Prints dog |
| 1500 | Moves dog |
| 7000 | Moves sheep and checks for end of game |
| 7180 | Prints sheep |
| 8000 | Scatters sheep |
| 9000 | Prints messages and end of game |

## Programming details

When you RUN this game you may imagine that there are some special techniques involved to govern the movement of the sheep

and sheepdog. This is, however, not the case and the program depends entirely on calculating their positions relative to each other according to mathematical equations. For example, line 7040 checks to see if the dog has approached too close to the sheep. If he has (or if the random number generated is less than .01, a one-in-a-hundred chance occurence) then the sheep scatter according to the equations in 8000 and 8010. IF statements are also used to make sure that the dog does not move into any of the picket fences or that the sheep do not move onto the dog or the fences.

## Scope for improvement

If you get really proficient at this game you can try to make it more difficult. You might increase the chance of the sheep scattering at random, by altering the value of the cut-off point for the random number in line 7050 or you could add some obstacles such as a pond or a river that the sheep has to avoid or cross.

## Program

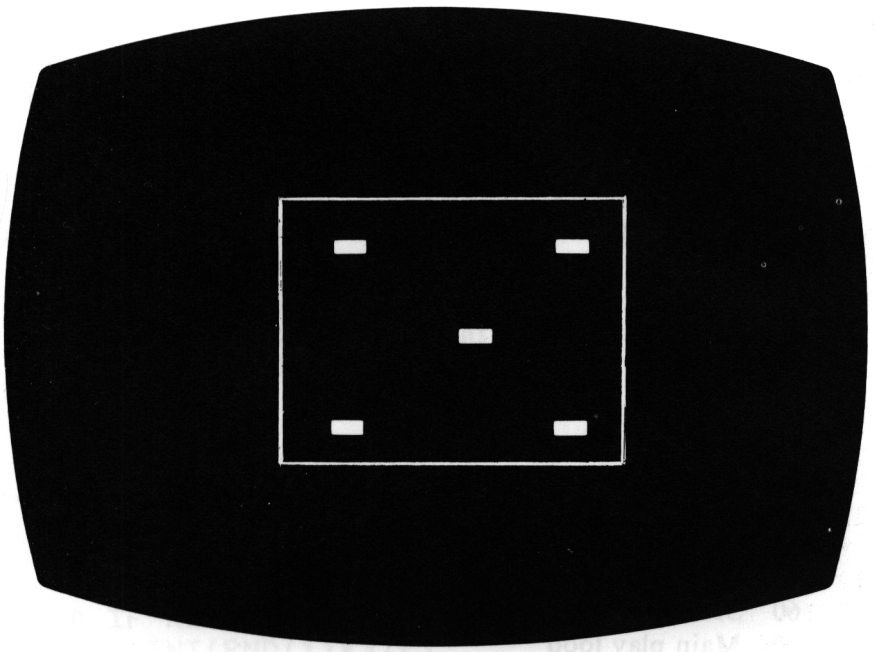
```
  10 REM Sheepdog Trial

 500 CHBAS=46080
 510 POKE CHBAS+(ASC("@")*8+0),0
 520 POKE CHBAS+(ASC("@")*8+1),0
 530 POKE CHBAS+(ASC("@")*8+2),26
 540 POKE CHBAS+(ASC("@")*8+3),63
 550 POKE CHBAS+(ASC("@")*8+4),63
 560 POKE CHBAS+(ASC("@")*8+5),61
 570 POKE CHBAS+(ASC("@")*8+6),36
 580 POKE CHBAS+(ASC("@")*8+7),36
 600 POKE CHBAS+(ASC("^")*8+0),0
 610 POKE CHBAS+(ASC("^")*8+1),1
 620 POKE CHBAS+(ASC("^")*8+2),3
 630 POKE CHBAS+(ASC("^")*8+3),29
 640 POKE CHBAS+(ASC("^")*8+4),28
 650 POKE CHBAS+(ASC("^")*8+5),36
 660 POKE CHBAS+(ASC("^")*8+6),18
 670 POKE CHBAS+(ASC("^")*8+7),0
 700 DIM Y(5)
 710 DIM X(5)
 720 M=0
```

```
 800 PAPER 2:INK 7
 810 CLS:PRINT CHR$(17)
 820 FOR X=1 TO 15
 830 PLOT X,16,"#"
 840 NEXT X
 850 FOR Y=0 TO 16
 860 PLOT 16,Y,"#"
 870 NEXT Y
 880 FOR Y=0 TO 20
 890 PLOT 0,Y,"#"
 900 PLOT 31,Y,"#"
 910 NEXT Y
 920 FOR X=0 TO 31
 930 PLOT X,0,"#"
 940 PLOT X,21,"#"
 960 NEXT X
 970 FOR Y=1 TO 3
 980 PLOT 12,Y,"#"
 990 NEXT Y

1000 FOR S=1 TO 5
1010 Y(S)=5+INT(RND(1)*10)
1020 X(S)=4+INT(RND(1)*6)
1030 PLOT X(S),Y(S),"@"
1040 NEXT S

1100 YD=1+INT(RND(1)*3)
1110 XD=1+INT(RND(1)*6)
1120 PLOT XD,YD,"^"
```

```
1500 PING:GET D$
1510 IF D$<>"R" THEN GOTO 1600
1520 PLOT 3,18,"RESIGN Y/N":GET A$
1530 IF A$="Y" THEN GOTO 9080
1540 PLOT 3,18,"                "
1600 PLOT XD,YD," "
1610 IF ASC(D$)=8 AND SCRN(XD-1,YD)=32 THEN XD=XD-1
1620 IFASC(D$)=9 AND SCRN(XD+1,YD)=32 THEN XD=XD+1
1630 IF ASC(D$)=11 AND SCRN(XD,YD-1)=32 THEN YD=YD-1
1640 IF ASC(D$)=10 AND SCRN(XD,YD+1)=32 THEN YD=YD+1
1650 PLOT XD,YD,"^"
1660 M=M+1
1670 M$=STR$(M)
1675 IF ASC(LEFT$(M$,1))>57 THEN
     M$=RIGHT$(M$,LEN(M$)-1):GOTO 1675
1676 IF ASC(LEFT$(M$,1))<48 THEN
     M$=RIGHT$(M$,LEN(M$)-1):GOTO 1676
1680 PLOT 21,14,"Move "+M$
1700 GOSUB 7000
1710 IF F=0 THEN GOTO 9000
1720 GOTO 1500

7000 F=0
7010 FOR S=1 TO 5
7020 Y=Y(S)
7030 X=X(S)
7040 IF (ABS(X(S)-XD)<2 AND ABS(Y(S)-YD)<2) THEN
     GOSUB 8000:GOTO 7100
7050 IF RND(1)<.01 THEN GOSUB 8000
7060 IF ABS(X(S)-XD)>2+INT(RND(1)*2) THEN GOTO 7100
7070 IF ABS(Y(S)-YD)>2+INT(RND(1)*2) THEN GOTO 7100
7080 X(S)=X(S)+SGN(X(S)-XD)
7090 Y(S)=Y(S)+SGN(Y(S)-YD)
7100 IF X(S)<1 THEN X(S)=1
7110 IF X(S)>14 THEN X(S)=14
7120 IF Y(S)<1 THEN Y(S)=1
7130 IF Y(S)>14 THEN Y(S)=14
7140 IF X(S)=12 AND Y(S)<4 THEN X(S)=11
7160 IF SCRN(X(S),Y(S))=35 THEN X(S)=X:Y(S)=Y:
     GOTO 7200
7170 IF SCRN(X(S),Y(S))=94 THEN X(S)=X:Y(S)=Y:
     GOTO 7200
```

```
7180 PLOT X,Y," "
7190 PLOT X(S),Y(S),"@"
7200 IF X(S)>12 AND Y(S)<4 THEN GOTO 7400
7300 F=1
7400 NEXT S
7410 RETURN

8000 X(S)=X(S)+(SGN(RND(1)-.5))*(2+INT(RND(1)*2))
8010 Y(S)=Y(S)+(SGN(RND(1)-.5))*(2+INT(RND(1)*2))
8020 RETURN

9000 IF M<40 THEN PRINT "SUPER SHEPHERD":
     GOTO 9080
9010 IF M<60 THEN PRINT "GOOD DOG":GOTO 9080
9020 IF M<90 THEN PRINT "KEEP PRACTISING":
     GOTO 9080
9030 IF M<120 THEN PRINT "LEARN TO WHISTLE":
     GOTO 9080


9040 PRINT "HAND IN YOUR CROOK"
9080 PRINT "YOU TOOK ";M;"MOVES"
9085 PRINT CHR$(17)
9090 INPUT "ANOTHER GAME Y/N";A$
9100 IF A$="Y" THEN RUN
9110 PAPER 7
9120 INK 0
```

# 2
# Magic Dice



Before the days of the micro, family games usually meant one of two things – card games or board games that involved dice. In our family we often couldn't find the dice and we spent ages hunting through drawers and cupboards before we could start our game. Equally often, in the excitement of the game, the dice would end up rolling over the floor and our game would be interrupted as we retrieved it from dark corners.

You may think there's no place for your old games of Ludo and Monopoly now you have a Oric to absorb you, but think again. They are actually very enjoyable games for lots of players especially if you don't have to spend too much time hunting for the dice, or worse still, arguing about which way up it actually fell! Such problems can be solved if you let your Oric join in the game and take over from the dice.

Of course, your Magic Dice can become the centre of a game. You can devise gambling games to play against the computer or against

other people. After all, dice have been around for thousands of years, so there must be plenty of ideas about how to use them.

However you choose to use the program, it will give you a large clear display on the TV screen – colourful too if you run it on a colour set. Notice the realistic way the dice actually slows down before it comes to a final halt and the use of sound effects to emphasize this feature.

## How to use the program

Using this program is simplicity itself. Type RUN and, when your Oric prompts, just press any key in order to start the dice rolling and then it carries on rolling for a random number of turns and slows down and stops when it is ready. The tone that sounds when the dice has stopped is longer than the one that accompanies each turn. When you've finished with the program press CTRL and C to stop it running.

## Subroutine structure

| | |
|---|---|
| 20 | Selects mode and disables cursor |
| 60 | Set-up routine |
| 100 | Main play loop |
| 260 | Prints and unprints dots |
| 430 | Draws yellow square for dice |
| 590 | Emits beep sound |

## Programming details

The essence of a dice program is in generating random numbers. In fact, this program uses random numbers in two ways. Firstly, randomness is used in the conventional way, to determine which face of the dice will show at the next turn – this is done in line 190, which uses the RND function to select 'R', a number between one and six. This information is then used in the printing subroutine (starting at line 260). The program goes to one of the six line numbers 280, 300, 330, 350, 380 and 400, according to the value of 'R', obeying a *computed* GOTO instruction which has the syntax

ON (result of arithmetic expression) GOTO

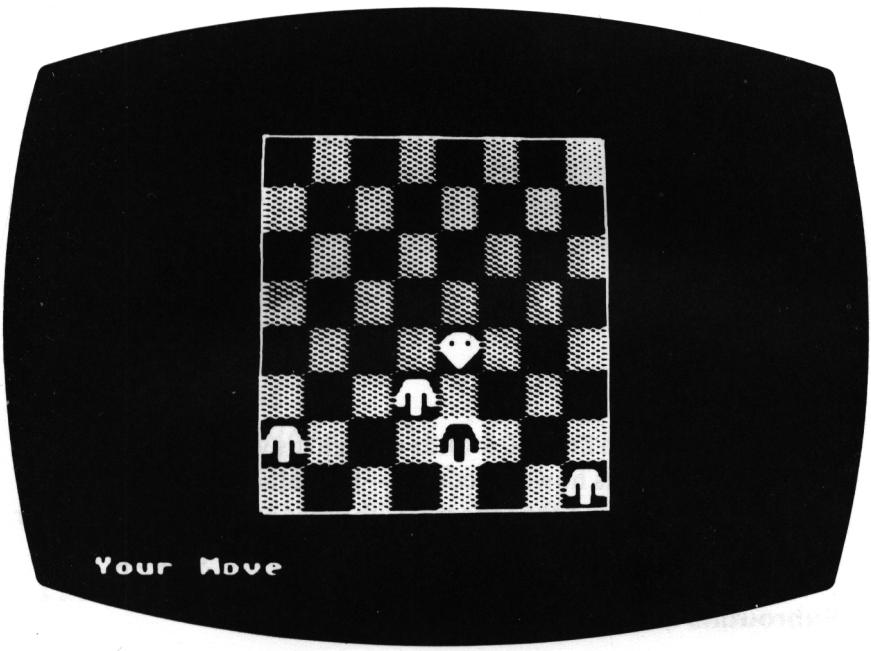
At 280 one dot is printed, at 300 two dots are printed and so on. The other use of the random number generator is to give the dice realistic suspense. When a human throws a dice it will turn just a few times or quite a number of times, and before it actually stops it will slow down. This program copies both these features by incorporating lines 130 and 140–220. Another random number, 'T', with a value between 5 and 8 is selected. This governs the numbers of turns the dice makes and each time it rolls over the pause before the dots re-appear lengthens.

### Program

```
 10 REM Dice

 20 PAPER 4
 30 INK 3
 40 CLS
 50 HIRES

 60 GOSUB 430
 70 PRINT CHR$(6);
 80 PRINT "PRESS ANY KEY TO THROW"

100 IF KEY$="" THEN GOTO 100
130 T=INT(RND(1)*3)+5
140 FOR I=1 TO T
160 A=1
180 GOSUB 260
190 R=INT(RND(1)*6)+1
200 A=0
210 GOSUB 260
220 NEXT I
240 PING
250 GOTO 100
```

```
260 GOSUB 590
270 ON R GOTO 280,300,330,350,380,400
280 CURSET 73,80,1
290 CHAR 120,1,A
295 RETURN
300 CURSET 50,60,1
305 CHAR 120,1,A
310 CURSET 98,100,1
320 CHAR 120,1,A
325 RETURN
330 GOSUB 280
340 GOTO 300
350 CURSET 50,100,1
355 CHAR 120,1,A
360 CURSET 98,60,1
365 CHAR 120,1,A
370 GOTO 300
380 GOSUB 350
390 GOTO 280
400 CURSET 73,60,1
405 CHAR 120,1,A
410 CURSET 73,100,1
415 CHAR 120,1,A
420 GOTO 350

430 REM SETUP
440 R=1
450 CURSET 50,50,0
460 FILL 70,10,127
480 PAPER 4
490 INK 3
500 SOUND 4,5,15
510 GOTO 260


590 SOUND 4,5,15
600 PLAY 0,1,0,0
610 WAIT 1
620 PLAY 0,0,0,0
630 WAIT I*2
640 RETURN
```

# 3
# Capture the Quark




What on earth is a 'quark'? Well may you ask that question but to find out you'll have to play this game. Here are just a few clues. The game is played on an eight-by-eight checkered board and the object of the game is to trap the quark and prevent him from reaching the bottom of the board. To do this you have two, three or four (determined at random) pieces, or 'quatlins' which can be moved diagonally one square at a time and only up the board. The quark also moves diagonally but he can move both forwards and backwards.

## How to play

At the beginning of the game your quatlins (two, three or four of them according to the luck of the draw) are ranged along the bottom line and the quark is at the top of the board. It is your move first.

You'll notice that one of your quatlins is displayed in reverse colours from the rest. This is the piece that is ready to move. If you want to move another piece hit any key and *control* will pass to the next piece in an anti-clockwise direction. Try pressing keys to see this in operation. When you are ready to move a piece press the left arrow key if you want to move diagonally forward left and the right arrow key if you want to move diagonally forward right. Once you have moved the quark will make his move automatically and its your turn again. The Oric will display a message when the game is won, either by you or the quark but if you want to resign before this, type "R". Just in case you hit this key by mistake you will be given the chance to reconsider and will have to answer "Y" or "N" to the question "Resign?"

## Typing tips

The IF statement in line 130 look as though it's wrong as there are no relational signs. It is however correct – the values stored in the array are either '1' or '0' and the Oric treats these as equivalent to 'true' or 'false'.

## Subroutine structure

| | |
|---|---|
| 20 | Initialises variables |
| 50 | Sets up game |
| 90 | Main play loop |
| 120 | Quark move logic |
| 500 | Moves quatlins and validates their moves |
| 650 | Selects which quatlin to move |
| 830 | Prints reversed quatlin |
| 890 | Prints quatlin |
| 930 | Draws board |
| 1120 | Draws initial positions of quatlins and quark and initialises board |
| 1390 | Defines graphics characters |
| 1590 | End of game |

## Programming details

Notice the way that subroutine 830 plots a quatlin in reverse by adding 128 to its ASCII code before plotting it. Both the quatlins and the quark are made up from four different user-defined graphics characters.

## Program

```
 10 REM Capture the Quark

 20 DIM D(10,10)
 30 DIM X(4),Y(4)
 40 DIM A(4),B(4)

 50 GOSUB 1390
 60 GOSUB 930
 70 H=INT(RND(1)*3)+2
 80 GOSUB 1120

 90 GOSUB 650
100 GOSUB 120
110 GOTO 90
```

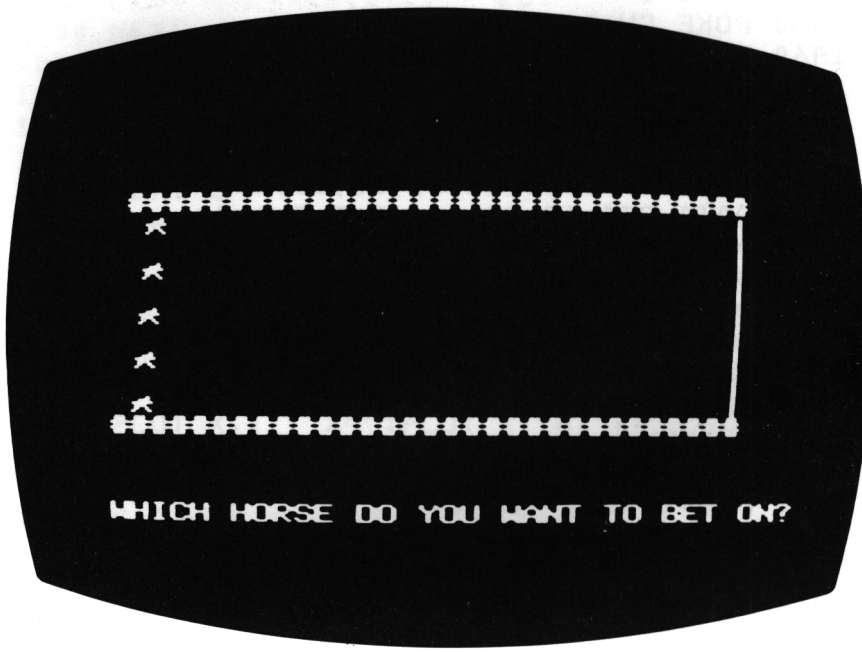
```
120 PLOT 1,21,"Quarks Move"
130 IF D(QI+2,QJ+2) AND D(QI+2,QJ) AND D(QI,QJ+2)
    AND D(QI,QJ) THEN GOTO 1590
140 M=1
150 GOSUB 400
170 IF QI+N<1 OR QI+N>8 THEN GOTO 200
180 IF D(QI+N+1,QJ+M+1) THEN GOSUB 330
190 IF D(QI+N+2,QJ+M+2)=1 AND D(QI+N,QJ+M+2)=1
    AND QJ<7 AND QJ>1 THEN M=-M
200 IF D(QI+N+1,QJ+M+1) THEN GOSUB 330
210 IF OI=QI AND OJ=QJ THEN M=-M:N=SGN(RND(1)-.5)
    :GOTO 170
220 OI=QI:OJ=QJ
230 PLOT QX,QY,"   ":PLOT  QX,QY+1,"  "
240 QX=QX+N*2
250 QY=QY+M*2
260 PLOT QX,QY,"!@":PLOT QX,QY+1,"#$"
270 D(QI+N+1,QJ+M+1)=2
280 D(QI+1,QJ+1)=0
290 QI=QI+N
300 QJ=QJ+M
310 IF QJ=8 THEN GOTO 1610
320 RETURN
330 N=-N
340 IF D(QI+N+1,QJ+M+1)<>1 THEN RETURN
350 M=-M
360 IF QJ<4 THEN N=1
370 IF D(QI+N+1,QJ+M+1)<>1 THEN RETURN
380 N=-N
390 RETURN
400 N=(QI<5)-(QI>=5)
410 R=RND(1)
420 IF QJ>6 THEN RETURN
430 IF R>.5 AND QI>3 THEN GOTO 470
440 IF QI>7 THEN GOTO 460
450 IF (D(QI+3,QJ+3)=0 OR D(QI+2,QJ+3)=0)
    AND D(QI+2,QJ+2)=0 THEN N=1:RETURN
460 IF QI<4 OR R>.5 THEN RETURN
470 IF (D(QI-3,QJ+3)=0 OR D(QI-2,QJ+3)=0)
    AND D(QI-2,QJ+2)=0 THEN N=1:RETURN
480 IF R>.5 THEN GOTO 440
490 RETURN
```

```
500 A(HM)=A(HM)+M
510 B(HM)=B(HM)-1
520 IF D(A(HM)+1,B(HM)+1)<>0 THEN GOTO 560
530 D(A(HM)-M+1,B(HM)+2)=0
540 D(A(HM)+1,B(HM)+1)=1
550 GOTO 600
560 A(HM)=A(HM)-M
570 B(HM)=B(HM)+1
580 PING
590 GOTO 660
600 PLOT X(HM),Y(HM),"  ":
    PLOT X(HM),Y(HM)+1,"  "
610 Y(HM)=Y(HM)-2
620 X(HM)=X(HM)+M*2
630 GOSUB 830
640 RETURN

650 PLOT 1,21,"Your move     "
660 PING
670 GET M$
700 IF M$="R" THEN GOTO 750
710 IF ASC(M$)=8 THEN M=-1:GOTO 500
720 IF ASC(M$)=9 THEN M=+1:GOTO 500
730 GOSUB 780
740 GOTO 650
750 INPUT "Resign Y/N";A$
760 IF A$="Y" THEN GOTO 1610
770 GOTO 650
780 GOSUB 890
790 HM=HM+1
800 IF HM>H THEN HM=1
810 GOSUB 830
820 RETURN

830 PLOT X(HM),Y(HM),ASC("%")+128:
    PLOT X(HM)+1,Y(HM),ASC("^")+128
850 PLOT X(HM),Y(HM)+1,ASC("&")+128:
    PLOT X(HM)+1,Y(HM)+1,ASC("*")+1
880 RETURN

890 REM PRINT QUATLIN
910 PLOT X(HM),Y(HM),"%^":
    PLOT X(HM),Y(HM)+1,"&*"
920 RETURN
```

```
 930 CLS
 940 K=2
 950 FOR I=1 TO 4
 960 FOR J=1 TO 4
 970 PLOT 4+J*4,K+I,"  CC"
 980 PLOT 4+J*4,K+I+1,"  CC"
1000 NEXT J
1010 K=K+2
1020 FOR J=1 TO 4
1030 PLOT 4+J*4,K+I,"CC  "
1040 PLOT 4+J*4,K+I+1,"CC  "
1050 NEXT J
1060 K=K+1
1070 NEXT I
1110 RETURN
1120 REM START
1130 FOR I=1 TO H
1140 X=I*4+6
1150 PLOT X,17,"%^":PLOT X,18,"&*"
1160 D(I*2+1,9)=1
1170 X(I)=X
1180 Y(I)=17
1190 HM=1
1200 A(I)=I*2
1210 B(I)=8
1220 NEXT I
1230 GOSUB 830
1240 QI=5
1250 QJ=1
1260 QX=QI*2+6
1270 QY=3
1280 PLOT QX,QY,"!@":PLOT QX,QY+1,"#$"
1290 FOR I=1 TO 10
1300 D(I,1)=1
1310 D(1,I)=1
1320 D(10,I)=1
1330 D(I,10)=1
1340 NEXT I
1350 D(QI+1,QJ+1)=2
1360 OI=0
1370 OJ=0
1380 RETURN
```

```
1390 REM INIT
1400 CH=46080
1410 FOR Q=1 TO 9
1420 READ C
1430 FOR I=0 TO 7
1440 READ D
1450 POKE CH+C+I,D
1460 NEXT I
1470 NEXT Q
1480 DATA 728,42,21,42,21,42,21,42,21
1490 DATA 264,0,0,7,15,63,51,51,63
1500 DATA 512,0,0,56,62,63,51,51,63
1510 DATA 280,63,63,31,15,7,3,1,0
1520 DATA 288,63,63,62,60,56,48,32,0
1530 DATA 296,0,0,15,31,31,63,59,51
1540 DATA 304,51,51,51,51,3,3,3,0
1550 DATA 752,0,0,60,62,62,63,55,55
1560 DATA 336,51,51,51,51,48,48,48,0
1570 CLS:PRINT CHR$(17);
1580 RETURN

1590 PLOT 1,21,"You win       "
1600 GOTO 1620
1610 PLOT 1,21,"Quark wins     "
1620 PRINT CHR$(17);:INPUT "Another game  ";A$
1630 IF A$="Y" THEN RUN
1640 CLS
1650 STOP
```

# 4
# Oric Ledger




This is a very simple betting game with an impressive and convincing horse race display plus an appropriate musical accompaniment. The tune is 'Camptown Races' and if you've ever heard it before you're sure to recognise it. If you want to show off the graphics and sound capabilities of your Oric this program provides a good demonstration.

### How to play

At the beginning of the game you are allocated a hundred chips. You have to bet on which horse will come in first and must decide how much to stake. The odds are 5 to 1, so if you win having placed 20 you will receive 100. The Oric keeps a tally of your winnings and losses and will tell you if you go broke. During the race, your horse is the white one but as the race is run automatically and at random so there's nothing you can do to make your horse win except cheer it along!

## Typing tips

Pay very careful attention when entering the numbers in the DATA statements in subroutine 6000. If you make mistakes in typing in this subroutine, the program may still run but the tune may be unrecognisable or sound discordant! The character between quotes in lines 1070 and 1080 is the hash symbol (which you produce by typing SHIFT and 3 together). You'll find the Oric's edit facility useful for entering lines 510 to 580 which all start in a similar way.

## Subroutine structure

| | |
|---|---|
| 500 | Defines graphics character for horse |
| 600 | Set up routine |
| 1060 | Prints race course |
| 1500 | Prints horses |
| 1550 | Runs race |
| 2000 | Title frame |
| 3000 | Betting routine |
| 4000 | Moves horses |
| 5000 | Plays whole tune |
| 6000 | Plays one note of tune |
| 8000 | Win/lose routine |
| 8500 | Move cursor to position |
| 8600 | Removes control codes from string M$ |
| 9000 | End of game |

## Programming details

This is the only program in this collection that plays a tune so it is worth drawing your attention to the details of lines 6000–6080. Think of these DATA statements as being made up of a pair of values. The first in each pair relates to the pitch of the note and the second to the length of time it is sounded for. This second value is typically 10, 5 or 2 – representing a minim, a crochet or a semiquaver respectively. The number −99 crops up instead of a pitch value every so often. The effect of this is to cause a *rest*, or pause in the music. Line 6080 signals the end of the tune. After detecting 99,99 the program resets the DATA statements so that next time round the tune starts playing from the beginning. At the beginning of the game

the tune is produced by calling subroutine 5000 which plays all the notes one after the other. However, because the Oric cannot do two things at once – it can't play a note and move the horses – the notes of the tune are produced by a call directly to subroutine 6000, which plays one note in the tune and then moves the pointer to the next note so that the next time it is called the next note is played. To synchronize the sound and the movement, the horses are moved then subroutine 6000 is called to play a note, the horses are moved again, another note is played and so on. The result is a sequence of notes with longer than normal rests between them but, because it does not take much time to move the horses, you still get the impression of a tune being played. You can use this technique in other games but, if the amount of calculation that you have to do between calling each note becomes too long, you'll no longer be able to hear the tune.

Subroutine 8600 is used to remove the attribute codes that the Oric places in front of the string of digits returned by the STR$ function.

## Scope for improvement

If you get tired of 'Camptown Races' as the background music you can substitute any tune you like. You could use the same graphics for a horse race program in which the player controlled one horse and tried to beat the rest of the field.

## Program

```
 10 REM Ledger

500 CHBAS=46080
510 POKE CHBAS+(ASC("@")*8+0),0
520 POKE CHBAS+(ASC("@")*8+1),2
530 POKE CHBAS+(ASC("@")*8+2),10
540 POKE CHBAS+(ASC("@")*8+3),61
550 POKE CHBAS+(ASC("@")*8+4),28
560 POKE CHBAS+(ASC("@")*8+5),18
570 POKE CHBAS+(ASC("@")*8+6),33
580 POKE CHBAS+(ASC("@")*8+7),0
```

```
 600 TEXT
 610 GOSUB 5000
 620 SUM=100
1000 DIM X(5)
1010 DIM Y(5)
1030 CLS
1040 PAPER 2
1050 INK 0

1060 FOR X=0 TO 31
1070 PLOT X,1,"#"
1080 PLOT X,11,"#"
1090 NEXT X
1100 FOR Y=2 TO 10
1110 PLOT 31,Y,"|"
1120 NEXT Y

1500 FOR Y=1 TO 5
1510 X(Y)=2
1520 Y(Y)=Y*2
1530 PLOT X(Y),Y(Y),"@"
1540 NEXT Y

1550 GOSUB 3000
1560 GOSUB 4000
1570 GOTO 1560

2000 INK 7
2010 PAPER 2
2020 PRINT CHR$(17)
2030 CLS
2040 PLOT 7,4,"O R I C  L E D G E R"
2050 RETURN
```

```
3000 PLOT 1,15, "WHICH HORSE DO YOU WANT TO BET ON
3005 ROW=15:COL=33:GOSUB 8500:INPUT B
3010 IF B<1 OR B>5 THEN PLOT 1,16,"NO SUCH HORSE"
     :GOTO 3000
3015 PLOT 1,Y(B),7
3020 M$="YOU HAVE "+STR$(SUM):GOSUB 8600
3025 PLOT 1,17,M$
3026 COL=30:ROW=18:GOSUB 8500
3030 PLOT 1,18, "HOW MUCH DO YOU WANT TO BET"
     :INPUT M
3040 IF SUM-M<0 THEN PLOT 1,19, "YOU DON'T
      HAVE THAT MUCH":GOTO 3026
3050 SUM=SUM-M


4000 Z=INT(RND(1)*5)+1
4010 PLOT X(Z),Y(Z)," "
4050 X(Z)=X(Z)+(1+RND(1)*.6)
4060 PLOT X(Z),Y(Z),"@"
4070 IF X(Z)>30 THEN GOTO 8000
4080 GOSUB 6000
4090 IF T=99 THEN RESTORE
4100 RETURN

5000 GOSUB 2000
5020 I=1
5030 GOSUB 6000
5040 IF T=99 THEN RESTORE:RETURN
5050 PLOT I,15," @"
5060 FOR Q=1 TO 10:NEXT Q
5070 I=I+.6
5080 GOTO 5030
```

```
6000 DATA 9,5,9,5,9,5,6,5,9,5,11,5,9,5,6,5,-99,
     5,6,5,4,15,6,5,4,10
6010 DATA 9,5,9,5,6,5,9,5,11,5,9,5,6,5
6020 DATA -99,5,6,2,4,2,2,2,4,2,6,5,4,5,2,15
6030 DATA -99,10,2,7,2,2,6,5,9,5,14,15
6040 DATA -99,5,11,7,11,2,14,5,11,5,9,15
6050 DATA 6,2,7,2,9,5,9,5,6,2,6,2
6060 DATA 9,2,9,2,11,5,9,5,6,10
6070 DATA 4,5,6,5,7,2,6,5,4,2,4,2,2,15
6080 DATA 99,99
6090 READ P,T
6100 IF T=99 THEN RETURN
6120 IF P=-99 THEN WAIT T*5:RETURN
6130 IF P>12 THEN MUSIC 1,4,P-12,0:GOTO 6150
6140 MUSIC 1,3,P,0
6150 PLAY 1,0,3,1500
6160 WAIT T
6170 PLAY 0,0,0,0
6180 RETURN

8000 IF Z=B THEN M$="YOU WIN "+STR$(INT(5*M))
     :SUM=SUM+INT(5*M)
8020 IF Z<>B THEN M$="YOU LOSE "+STR$(M)
8030 IF SUM<=0 THEN M$="YOU'RE BROKE"
     :GOSUB 8600:PLOT 1,23,M$:STOP
8040 GOSUB 8600
8050 PLOT 1,22,M$
8060 GOTO 9000

8500 DOKE #12,48040+ROW*40+COL
8510 POKE #268,ROW+1
8520 RETURN

8600 J=1
8610 IF J>LEN(M$) THEN RETURN
8620 IF ASC(MID$(M$,J,1))>31 THEN J=J+1:GOTO 8610
8630 M$=LEFT$(M$,J-1)+MID$(M$,J+1)
8640 GOTO 8610
```

```
9000 M$="YOU HAVE "+STR$(SUM)+" ANOTHER RACE Y/N
9010 GOSUB 8600:PLOT 1,23,M$
9015 GET A$
9020 RESTORE
9030 IF A$="Y" THEN GOTO 1030
9040 IF A$<>"N" THEN GOTO 9000
9050 PAPER 7
9060 PRINT CHR$(17)
9070 CLS
9080 STOP
```

# 5
# Across the Ravine


Have you got the skill and judgement needed to lead an expedition through dangerous terrain? This colour graphics game provides an easy way to discover your potential. The object of the game is to get your party of five intrepid explorers across a deep ravine with a fast flowing river at the bottom of steep cliffs. There is only one option, to swing across on a rope. The rope swings all the time so each man must run and leap to catch it – anyone who mistimes his jump falls into the river and is lost. Listen out for the sound effects as a man falls towards the river! The Oric's EXPLODE command provides a realistic splash sound.

## How to play

This game is all a matter of timing – you have to judge when to start each run for the rope. It is vital to catch the rope on the downswing

and each man can jump approximately his own width. You can wait as long as you like before making any run and when you are ready press any key to jump. When your first little man has made his run and either swung to safety or perished the second explorer will appear in position. There are five of them in all and the length of the rope remains the same for all of them. With each new game the rope changes at random.

## Typing tips

Remember that you can save effort by taking advantage of the Oric's copy facility when inputting lines that are similar to earlier ones. Use the cursor keys to move to the characters you want to copy and then press CTRL and A together.

In line 740 there are two sets of quotation marks with nothing in between. This is known as a *null string*.

## Subroutine structure

| | |
|---|---|
| 20 | Initialises arrays |
| 40 | Main play loop |
| 110 | Swings rope |
| 310 | Man across routine |
| 400 | Calculates positions of rope |
| 470 | Prints ravine |
| 680 | Plots man on end of rope |
| 740 | Man jumps routine |
| 860 | Gets next man ready |
| 980 | Man falls in water routine |
| 1150 | Sets up game |
| 1240 | End of game |

## Programming details

This program uses an interesting combination of low and high resolution graphics. The river banks are created using the FILL command. The rope is plotted in high resolution grahpics and its old positions are blanked out by re-plotting them. The little man figure is a low resolution user-defined graphic produced in high resolution

mode using the CHAR command. This means that he can appear anywhere on the screen. He therefore seems to move smoothly rather than in the jerky fashion that would result if he could only be printed at set character positions.

Another point to note is the way the co-ordinates of the positions of the swinging rope are first calculated by subroutine 400 and stored in two arrays, 'X' and 'Y'. These positions are then used repeatedly in the plotting of the swinging rope. This saves having to recalculate them each time they are needed and so speeds the whole program up. This technique can be applied to any situation where anything is moving rhythmically or periodically. For another example, see Mighty Missile.

### Scope for alteration

You can make the game easier by increasing the value to the right of the "<" in line 830 or make it more difficult by decreasing this value.

### Program

```
 10 REM Across the Ravine

 20 DIM X(32)
 30 DIM Y(32)

 40 GOSUB 470
 50 GOSUB 400
 60 GOSUB 1150
 70 GOSUB 860
 80 GOSUB 110
 90 IF MEN=0 THEN GOTO 1240
100 GOTO 80
```

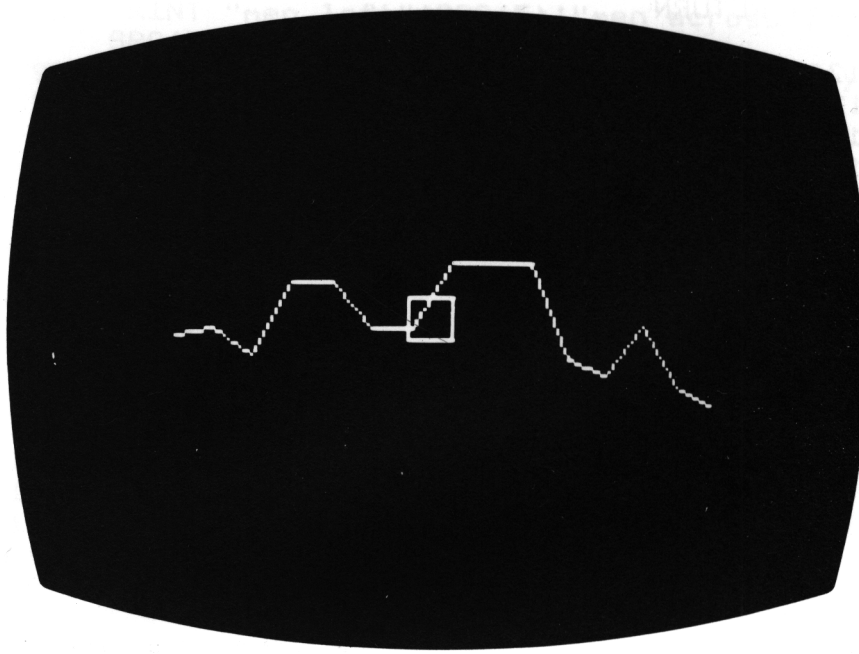
```
110 FOR T=1+R TO N-R
130 CURSET 118,40,1
140 DRAW X(T),Y(T),1
150 S=1:GOSUB 680
160 CURSET 118,40,0
170 DRAW X(T),Y(T),0
180 IF J=1 THEN S=2:GOSUB 680
190 NEXT T
200 IF C=1 THEN GOTO 310
210 FOR T=N-R TO 1+R STEP -1
230 CURSET 118,40,0
240 DRAW X(T),Y(T),1
250 S=3:GOSUB 680
260 CURSET 118,40,0
270 DRAW X(T),Y(T),0
280 IF J=1 THEN S=4:GOSUB 680
290 NEXT T
300 RETURN

310 ACROSS=ACROSS+1
320 DX=(ACROSS-1)*10+15
330 DY=140
340 GOSUB 710
350 C=0
360 MEN=MEN-1
370 IF MEN=0 THEN GOTO 920
380 GOSUB 860
390 RETURN

400 N=0
410 FOR T=-PI/6 TO PI/6 STEP .05
420 N=N+1
430 X(N)=INT(-100*SIN(T))
440 Y(N)=INT(100*COS(T))
450 NEXT T
460 RETURN
```

```
470 TEXT:PAPER 4:INK 7
475 PRINT CHR$(6)
480 CH=46080
490 C=ASC("@")*8
500 FOR I=0 TO 7
510 READ D
520 POKE CH+C+I,D
530 NEXT I
540 DATA 12,12,63,63,12,18,51,51
545 HIRES:PRINT CHR$(17)
560 CURSET 8,150,3
570 FILL 50,15,127
580 CURSET 150,150,3
590 FILL 50,15,127
600 C=0
610 J=0
620 INK 2
630 PAPER 7
670 RETURN

680 IF C=0 THEN GOTO 740
690 DX=115+X(T)
700 DY=Y(T)+41
710 CURSET DX,DY,3
720 CHAR 64,0,2
730 RETURN

740 IF KEY$="" AND J=0 THEN WAIT 25:RETURN
750 J=1
760 DY=MY
770 DX=MX
780 GOSUB 710
790 MX=MX-8
800 DY=MY
810 DX=MX
820 GOSUB 710
830 IF ABS(MX-118-X(T))<10 AND S=2 THEN C=1:
    GOTO 710
840 IF MX<150 THEN GOTO 980
850 RETURN
```

```
 860 MX=230
 870 MY=140
 880 DY=MY
 890 DX=MX
 900 J=0
 910 GOSUB 710
 920 PRINT "men left";SPC(5);"men across";
     SPC(5);"men lost"
 930 PRINT SPC(4);MEN;SPC(12);ACROSS;SPC(11);LOST
 940 IF MEN=0 THEN GOTO 1240
 960 RETURN


 980 GOSUB 710
 985 F=1
 990 MY=MY+8
1000 DY=MY
1010 GOSUB 710
1020 MUSIC 1,5,F,10
1025 PLAY 1,0,0,0
1030 WAIT 2
1040 GOSUB 710
1050 IF MY<185 THEN F=F+1:GOTO 990
1060 EXPLODE
1070 LOST=LOST+1
1080 C=0
1090 J=0
1100 MEN=MEN-1
1110 IF MEN=0 THEN GOTO 920
1120 WAIT 200
1130 GOSUB 860
1140 RETURN
```

```
1150 MEN=5
1160 LOST=0
1170 ACROSS=0
1180 J=0
1190 C=0
1200 R=INT(RND(1)*5)
1230 RETURN

1240 PING:WAIT 100
1245 TEXT:PRINT CHR$(6)
1250 PRINT "You lost "LOST
1260 INPUT "Another game Y/N ";A$
1270 IF A$="Y" THEN RUN
1280 CLS
1290 STOP
```

# 6
# Guideline



This is a game of skill in which you have to guide a square along an irregular wavy wire. When you play this game with a real wire and a ring, it's a test of how steadily you can guide the ring along the wire. In this Oric version it's a matter of speed as well as hand and eye co-ordination. The square moves from left to right across the screen automatically but you have to keep it on course by pressing the up and down arrow keys. The object of the game is to be on target for as much of the length of the wire as possible and at the end of the game the percentage of time you were successful is displayed.

## How to play

At the beginning of the game you have to select the level of difficulty of the game – from 1 (easy) to 5 (difficult). This determines the size of the square that has to be guided along the wire. Once the square

appears, a tone sounds to indicate the start of the game and then the square automatically starts to move right. Press the up and down arrow keys to change its direction to ensure that it stays on the wire. Keeping your finger down on an arrow key will keep the square moving in the same direction. If the square leaves the wire another tone will sound and will continue to sound at every move until you regain a position on the wire.

### Subroutine structure

20 Initialisation routine
140 Call to main play loop
160 End of game
250 Initialises variables and prints title frame
480 Plots wire
590 Draws and moves square and calculates amount on target
840 Off target sound

### Programming details

High resolution graphics are used in this program to give the smooth movement needed to test the players' skill. Subroutine 480 is responsible for plotting the line for the wire and the square is constructed by a collection of PLOT statements at the beginning of subroutine 590. The POINT function is used in this subroutine to test whether the square ring is actually on target around the wire.

Notice the way that the ASC function is used in lines 660 and 670 to discover if the up and down arrow keys have been pressed. The up arrow corresponds to an ASCII code of 10 and the down arrow to an ASCII code of 11.

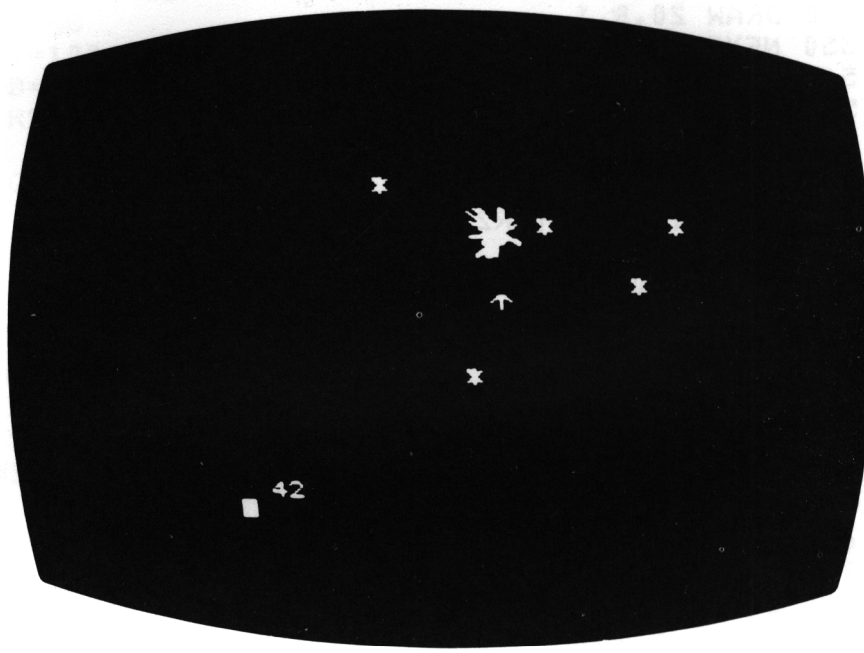
### Program

```
10 REM Guideline
```

```
 20 PRINT CHR$(17)
 30 GOSUB 250
 40 PRINT CHR$(6)
 70 GOSUB 480
 80 WAIT 100
 90 PRINT "GO"
100 PING
110 Y=100:X=20
120 B=Y:R=30-DF
130 R2=R/2:V=3

140 GOSUB 590

160 PRINT "You were on target "
170 PRINT INT(HIT/(HIT+MISS)*10000)/100;
    "% of the time"
180 INPUT  "Another game Y/N";A$
190 PRINT CHR$(17);CHR$(6);
200 IF A$="Y" THEN RUN
210 IF A$<>"N" THEN GOTO 180
220 CLS
230 TEXT
240 STOP

250 X=10
260 Y=100
270 D=15
280 P=0
290 TEXT:CLS
300 PLOT 10,2,"G U I D E L I N E"
310 PLOT 2,4,"You must guide a ring along the"
320 PLOT 2,5,"wavy 'wire' using the up and"
330 PLOT 2,6,"down arrow keys"
340 PLOT 2,8,"You will be marked on how"
350 PLOT 2,9,"accurate you are"
360 PLOT 2,11,"Select the  difficulty level"
370 PLOT 2,12," 1 (easy) to 5 (difficult) "
380 GET DF
390 IF  DF<1 OR DF>5 THEN GOTO 380
400 DF=DF*4
450 HIRES
460 RETURN
```

```
480 CURSET X,Y,1
490 FOR I=1 TO 10
500 R=D-INT(RND(1)*(2*D)-1)
510 Y=Y+R
520 IF Y>180 THEN Y=Y-R:R=-R
530 IF Y<10  THEN Y=Y+R:R=-R
540 DRAW 20,R,1
550 NEXT I
560 HIT=0
570 MISS=0
580 RETURN

590 CURSET X-R2,Y-R2,2
600 DRAW 0,R,2
610 DRAW R,0,2
620 DRAW 0,-R,2
630 DRAW -R,0,2
640 A$=KEY$
650 IF  A$="" THEN GOTO  680
660 IF ASC(A$)=11 AND Y-R2>2*V THEN B=B-2*V
670 IF ASC(A$)=10 AND Y+R<190-2*V THEN B=B+2*V
680 CURSET X-R2,Y-R2,2
690 DRAW 0,R,2
700 DRAW R,0,2
710 DRAW 0,-R,2
720 DRAW -R,0,2
730 X=X+V
740 IF X>210 THEN RETURN
750 Y=B
760 F=0
770 FOR I=-R2+1 TO R2-1
780 IF POINT(X,Y+I)=-1 THEN HIT=HIT+1:
    WAIT 10:I=R-1:F=1
790 NEXT I
800 IF F=1 THEN GOTO 590
810 MISS=MISS+1
820 GOSUB 840
830 GOTO 590

840 SOUND 1,50,0
850 PLAY 1,0,1,2000
860 WAIT 1
870 PLAY 0,0,0,0
880 RETURN
```

# 7
# Laser Attack




This is a very exciting and fast moving space fight game that uses sound effects and some rather unusual graphics techniques to good effect. The screen is treated as if it were a spherical universe. So, if you go off the top of the screen you re-appear at the bottom and if you go off at the right you re-appear at the left. This game is a race against time. The Oric will count to two hundred while you try to annihilate the enemy ship with your infallible laser weapon – the chase is on.

## How to play

At the beginning of this game you have to select a difficulty factor. This governs the unpredictability of the enemy ship's course and the number of stars that appear. The stars act as obstacles in this game. If you hit one you will be deflected at random so the fewer there are

the easier it is to steer your course. Your ship moves continuously. It is shaped like an arrow-head and can point in any of eight directions. Every time you press any key it turns clockwise through 45 degrees. The enemy is a revolving cartwheel-shaped disc that meanders through space. To fire your laser press the space bar. Your weapon will fire in a straight line from the point of your arrow. If you hit the enemy ship it will disintegrate with appropriate sound and visual effects. The counter is constantly displayed at the bottom left of the screen and when it reaches 200 your time is up.

### Typing tips

Notice the null string in line 130 – there should not be a space between the two sets of quotes marks.

### Subroutine structure

| | |
|---|---|
| 20 | Main play loop |
| 120 | Fires or rotates direction of arrow |
| 200 | Laser zap routine |
| 380 | Explosion routine |
| 490 | Detects hit |
| 560 | Moves arrow |
| 670 | Moves target |
| 840 | Title frame |
| 950 | Gets and prints time |
| 990 | Prints stars |
| 1040 | Initialises variables and defines graphics |
| 1400 | Draws arrow |
| 1890 | End of game |

### Programming details

This is a complicated program and one that illustrates a number of novel programming methods. The arrow is produced by a collection of CURSET and DRAW commands. The cartwheel is produced by printing one of two user-defined characters on the screen using the CHAR command. The arrays V and W are used to hold the horizontal and vertical velocities that determine the movement of

both of them. They also determine the direction in which the lines that make up the moving arrow are drawn. This technique ensures that the arrow keeps pointing the same way it is moving.

## Scope for improvement

If you feel very ambitious you could make this game even more exciting by enabling the enemy ship to fire at random – so that you have to dodge its fire at the same time as pursuing it.

## Program

```
 10 REM Laser Attack

 20 PRINT CHR$(17)
 30 GOSUB 840
 40 GOSUB 1040
 50 GOSUB 950
 60 IF INT(TIME)>200 THEN GOTO 1890
 70 GOSUB 120
 80 GOSUB 560
 90 IF H=1 THEN GOTO 1890
100 GOSUB 700
110 GOTO 50

120 A$=KEY$
130 IF A$="" THEN GOSUB 1400:RETURN
140 IF ASC(A$)=32 THEN GOTO 200
150 GOSUB 1400:K=K+1
160 IF K>8 THEN K=1
170 RETURN
```

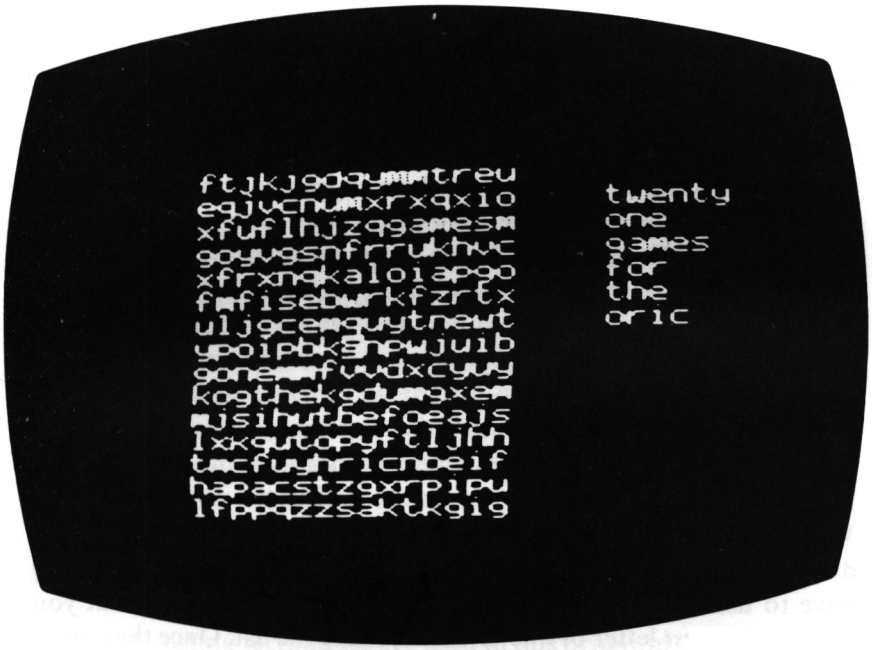
```
200 XL=X+6*V(K)
210 YL=Y+6*W(K)
220 GOSUB 490
230 CURSET XL,YL,0
240 DX=0
250 IF V(K)=1 THEN DX=230-XL
260 IF V(K)=-1 THEN DX=6-XL
270 DY=0
280 IF W(K)=1 THEN DY=190-YL
290 IF W(K)=-1 THEN DY=6-YL
300 IF V(K)*W(K)=0 THEN GOTO 330
310 IF ABS(DX)<ABS(DY) THEN DY=ABS(DX)*SGN(DY):
    GOTO 330
320 DX=ABS(DY)*SGN(DX)
330 DRAW DX,DY,1
340 CURSET XL,YL,0
350 ZAP
360 DRAW DX,DY,0
370 IF H=0 THEN GOSUB 1400:RETURN

380 MX=B:MY=A
390 FOR I=1 TO RND(1)*5+20
400 DX=10-INT(RND(1)*20+1)
410 IF MX+DX>230 OR MX+DX<6 THEN GOTO 470
420 DY=10-INT(RND(1)*20+1)
430 IF MY+DY>190 OR MY+DY<0 THEN GOTO 470
435 IF DY=0 AND DX=0 THEN GOTO 470
440 CURSET MX,MY,1
450 DRAW DX,DY,1
460 ZAP
470 NEXT I
480 GOTO 1900

490 H=0
500 DY=A-Y-6*W(K)
510 DX=B-X-6*V(K)
520 IF W(K)*DX<>V(K)*DY THEN RETURN
530 IF ABS(V(K))*SGN(DX)<>V(K) OR
    OR ABS(W(K))*SGN(DY)<>W(K) THEN RETURN
540 H=1
550 RETURN
```

```
560 REM MOVE ARROW
570 X=X+V(K)*6
580 Y=Y+W(K)*6
590 IF X<12 THEN X=228
600 IF X>228 THEN X=6
610 IF Y<12 THEN Y=180
620 IF Y>180 THEN Y=12
630 IF POINT(X,Y)=-1 THEN EXPLODE:GOTO 670
640 GOSUB 1400
660 RETURN

670 K=K+1
680 IF K>8 THEN K=1
690 GOTO 560
700 IF RND(1)>1.05-DF/20 THEN Z=Z+1
710 IF Z>8 THEN Z=1
720 CURSET B-3,A-3,0:CHAR 127,0,0
730 A=A+V(Z)*6
740 B=B+W(Z)*6
750 IF B<12 THEN B=228
760 IF B>228 THEN B=12
770 IF A<12 THEN A=180
780 IF A>180 THEN A=12
790 IF POINT(B,A)=-1 THEN SHOOT:Z=Z+1:GOTO 710
800 CURSET B-3,A-3,0:CHAR ASC(W$(R+1)),0,1
810 IF R=0 THEN R=1 ELSE R=0
830 RETURN

840 TEXT:PAPER 4:INK 7:CLS
850 PLOT 8,2,"L A S E R  A T T A C K"
860 PLOT 2,6,"You are in control of an advanced"
870 PLOT 2,7,"laser attack ship in pursuit"
880 PLOT 2,8,"of an enemy craft"
890 PLOT 2,10,"Shoot it down before your time
    is up"
900 PLOT 2,15,"Select the difficulty level"
910 PLOT 2,16,"1 (easy) to 9 (difficult)
920 GET DF
930 IF DF<1 OR DF>9 THEN GOTO 920
940 RETURN

950 TIME=TIME+1
960 PRINT:PRINT TIME
980 RETURN
```

```
 990 REM star
1000 IF RND(1)>.5+DF/50 THEN RETURN
1010 CURSET RND(1)*230,RND(1)*190,1
1020 CHAR 42,0,1
1030 RETURN

1040 DIM W(8),V(8)
1060 DATA -1,1,0,1,1,1,1,0,1,-1,0,-1,-1,-1,
          -1,0,-1,1
1070 FOR I=0 TO 8
1080 READ W,V:W(I)=W:V(I)=V
1090 NEXT I
1100 K=1
1110 X=60
1120 Y=30
1140 B=INT(RND(1)*10)*6+24
1150 A=INT(RND(1)*10)*6+24
1190 CH=46080
1200 FOR Q=1 TO 2
1210 READ C
1220 FOR I=0 TO 7
1230 READ D
1240 POKE CH+C*8+I,D
1250 NEXT I
1260 NEXT Q
1265 HIRES
1270 DATA 91,16,33,19,12,12,50,33,2
1280 DATA 93,7,36,36,60,15,5,5,28
1290 W$(1)="["
1300 W$(2)="]"
1310 R=0
1320 NB1=0
1330 NB2=0
1335 GOSUB 1400
1340 TIME=0
1345 FOR J=1 TO 10+INT(RND(1)*DF):GOSUB 990:
     NEXT J
1350 H=0
1355 PAPER 4
1360 RETURN
```

```
1400 CURSET X,Y,2
1410 DRAW 6*V(K),6*W(K),2
1420 I=K+3:I=I-INT(I/8)*8
1430 DRAW 3*V(I),3*W(I),2
1440 CURSET X+6*V(K),Y+6*W(K),2
1450 I=K+5:I=I-INT(I/8)*8
1460 DRAW 3*V(I),3*W(I),2
1470 RETURN

1890 IF H<>1 THEN GOTO 1930
1900 WAIT 100:TEXT
1910 PRINT "You did it "
1920 GOTO 1940
1930 PRINT "Your time is up"
1940 PRINT CHR$(17);CHR$(6)
1950 INPUT "Another game Y/N";A$
1960 IF A$="Y" THEN RUN
1970 CLS
1980 STOP
```

# 8
# Word Scramble




This is a game that all the family can play – and it really presents quite a challenge even to the most sharp-eyed and keen-witted. Your Oric invites you to give it a list of up to ten words, each up to ten letters long. Once you have entered them it hides them within a fifteen-by-fifteen grid and fills up all the vacant spaces with random letters. All the words you've entered appear along straight lines – vertically, horizontally or diagonally – but they can be backwards, upside-down, or slanting from bottom to top. Once they have been camouflaged by all the random letters spotting them is like looking for a needle in a haystack. If you want to make the task a little easier you can opt to preview the puzzle before any extra letters are added. This at least gives you a chance to unscramble the puzzle. There is yet another helpful hint you can opt for. You can have the list of hidden words displayed on the screen beside the puzzle – but you may be surprised how difficult to spot they still are.

The object of the game is to find all the hidden words. Your

position in the square is indicated by an inverted character which you can use as a pointer to identify the first letter of each word you find. When you think you have found the first letter of a word type a 'w'. If you are correct you score a point, otherwise you'll hear a dismal tone.

## How to play

The Oric guides you through the early stages of this game, asking you first how many words you wish to supply and then prompting you for each. You must use *lower case* only when supplying the words of your choice. Then it has to create the puzzle – which takes it a little time – longer the more long words you've included. It tells you that its "working" on it so that you don't think it's forgotten about you. When it's ready it asks if you want to preview the puzzle. If you prefer to play the game without any advantages you can skip the preview by answering "n". Similarly, you can answer either "y" or "n" to the next question which gives the option of having the list of hidden words displayed on the screen beside the completed grid. When the grid appears, you'll see that the top left hand position is displayed in inverted graphics. This is where the pointer starts. You have to use the arrow keys to move this cursor to a letter that you think is the *first* letter of one of the words in the list. Once the cursor is in position, type "w" – again using *lower case*. The Oric will then ask you for the word that you have identified. Type this in. If you are correct you will score one point (and your score total will go up by one) but if you are wrong you will hear a tone. Once you've completed the puzzle you'll see the message "You got them all". If you want to give up during the game type "r" and you'll be given the option to resign.

## Typing tips

When playing this game remember to use lower case letters only. There is a single space between double quotes in lines 410, 590, 800, 810, 930 and 1420 so don't type in a null string instead.

## Subroutine structure

| | |
|---|---|
| 40 | Initialisation routine |
| 80 | Calls finding words routine |
| 90 | Asks for words to be input |
| 290 | Gives error message if more than ten letters input |
| 330 | Finds longest word left in list |
| 380 | Constructs puzzle |
| 760 | Prints puzzle |
| 870 | Allows preview and fills up puzzle with random letters |
| 1010 | Main play loop |
| 1230 | Asks for guess of word |
| 1320 | Word correct routine |
| 1450 | Checks that word is in list |
| 1500 | Checks for word |
| 1620 | Defines dot character and sets score to zero |
| 1670 | Resign routine |
| 1715 | End of game |
| 2000 | Moves cursor to position |

## Programming details

Some interesting techniques are used in this program. The words are fitted into the square by choosing starting points at random (lines 500–510) and also by choosing directions at random (lines 520–530). Each word is then tested against the square position-by-position and if there is an empty space, or the identical letter is already present, for every letter of the word then the word goes in. This allows for two or more words to cross over sharing a space at a common letter. If the word can't be fitted in at its first random spot and direction, the program jumps back and chooses another spot and direction. This procedure is repeated until all the words are fitted.

## Scope for improvement

If you have a printer, you could add a routine to this program to enable the completed puzzle to be printed out so that you take it away to be solved. To make the game more difficult you could allow it to accept more words. Notice, however, that the more words there are the longer it will take to be set up initially. To make the game

easier you could remove words from the word list, or mark them in some way, once they had been found.

## Program

```
 10 REM Word Scramble

 40 GOSUB 90
 50 GOSUB 380
 60 GOSUB 1620
 70 GOSUB 870

 80 GOSUB 1010

 90 DIM L$(10)
100 DIM C(10)
110 LST=0
130 CLS
140 INPUT "How many words ";W
150 IF W<2 OR W>10 THEN PING:GOTO 140
170 PRINT "Enter words (lower case only)"
175 PRINT CHR$(20)
180 FOR I=1 TO W
190 PRINT "Word Number ";I;"=";
200 INPUT W$
210 IF LEN(W$)>10 THEN GOSUB 290
220 IF LEN(W$)<1 THEN GOTO 200
230 L$(I)=W$
240 C(I)=LEN(W$)
250 NEXT I
270 RETURN

290 PRINT "Maximum of ten letters"
300 INPUT W$
310 PRINT "Word Number ";I;"=";W$
320 RETURN

330 M=0:J=0
340 FOR Z=1 TO W
350 IF M<C(Z) THEN M=C(Z):J=Z
360 NEXT Z
370 RETURN
```

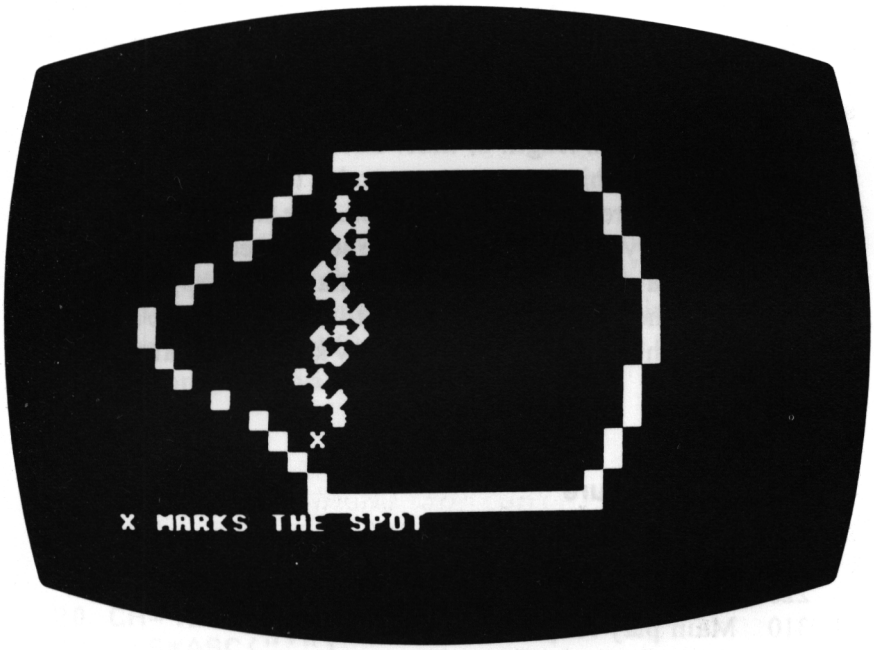
```
380 DIM D$(15,15)
390 FOR J=0 TO 14
400 FOR I=0 TO 14
410 D$(I,J)=" "
420 NEXT I
430 NEXT J
440 CLS
450 F=0
460 FOR I=1 TO W
470 GOSUB 330
480 L=C(J)
490 C(J)=0
500 X=INT(RND(1)*(15-L))+1
510 Y=INT(RND(1)*(15-L))+1
520 V=INT(RND(1)*3)-1
530 U=INT(RND(1)*3)-1
540 IF U=0 AND V=0 THEN GOTO 520
550 A=X:B=Y
560 IF V<0 THEN A=A+L
570 IF U<0 THEN B=B+L
580 FOR K=1 TO L
590 IF D$(A,B)<>" " AND D$(A,B)<>MID$((L$(J)),K,1)
    THEN F=1:K=L:GOTO 620
600 A=A+V
610 B=B+U
620 NEXT K
640 IF F=1 THEN F=0:GOTO 500
650 PRINT "working"
660 A=X:B=Y
670 IF V<0 THEN A=A+L
680 IF U<0 THEN B=B+L
690 FOR K=1 TO L
700 D$(A,B)=MID$(L$(J),K,1)
710 A=A+V
720 B=B+U
730 NEXT K
740 NEXT I
750 RETURN
```

```
 760 REM print
 770 CLS
 780 FOR M=0 TO 14
 790 FOR N=0 TO 14
 800 IF D$(N,M)=" " THEN PRINT "!";
 810 IF D$(N,M)<>" " THEN PRINT D$(N,M);
 820 NEXT N
 830 IF LST=0 OR M>10 THEN PRINT
 840 IF LST=1 AND M<=10 THEN PRINT "     ";L$(M)
 850 NEXT M
 860 RETURN

 870 PRINT "Do you want to preview"
 880 INPUT "the puzzle y/n";A$
 890 IF LEN(A$)=0 THEN GOTO 870
 900 IF A$="y" THEN GOSUB 760
 910 FOR I=0 TO 14
 920 FOR J=0 TO 14
 930 IF D$(J,I)=" " THEN D$(J,I)=CHR$(INT(RND(1)*
     26)+97)
 940 NEXT J
 950 NEXT I
 960 PRINT "Do you want to display"
 970 INPUT "the words beside the puzzle y/n";A$
 980 IF LEN(A$)=0 THEN GOTO 960
 990 IF A$="y" THEN LST=1
1000 GOSUB 760:RETURN


1010 X=0
1020 Y=0
1030 PRINT CHR$(17);
1050 PLOT X+1,Y,ASC(D$(X,Y))+128
1080 GET A$
1090 IF A$="w" THEN GOTO 1230
1100 PLOT X+1,Y,D$(X,Y)
1120 IF ASC(A$)=8 AND X>0 THEN X=X-1
1130 IF ASC(A$)=9 AND X<14 THEN X=X+1
1140 IF ASC(A$)=10 AND Y<14 THEN Y=Y+1
1150 IF ASC(A$)=11 AND Y>0 THEN Y=Y-1
1160 IF A$="r" THEN GOSUB 1670
1190 PLOT X+1,Y,ASC(D$(X,Y))+128
1220 GOTO 1080
```

```
1230 ROW=20:COL=2:GOSUB 2000:INPUT "What is
     the word";W$
1240 IF LEN(W$)=0 OR LEN(W$)>15 THEN PING:
     GOTO 1230
1250 GOSUB 1450
1255 ROW=20:COL=2:GOSUB 2000:PRINT SPC(35)
1280 IF MATCH=0 THEN PING:GOTO 1080
1290 FOR U=-1 TO 1
1300 FOR V=-1 TO 1
1310 IF U=0 AND V=0 THEN GOTO 1340

1320 GOSUB 1500
1330 IF MATCH=1 THEN V=1:U=1
1340 NEXT V
1350 NEXT U
1360 IF MATCH=1 THEN GOTO 1390
1370 PING
1375 ROW=20:COL=2:GOSUB 2000
1380 GOTO 1080
1390 SC=SC+1
1400 COL=2:ROW=22:GOSUB 2000
1405 PRINT "Score= ";SC
1410 ZAP
1420 L$(WO)=" "
1430 IF SC=W THEN GOTO 1770
1440 GOTO 1080

1450 MATCH=0
1460 FOR I=1 TO W
1470 IF W$=L$(I) THEN MATCH=1:WO=I:I=W
1480 NEXT I
1490 RETURN

1500 REM MATCH
1510 MATCH=1
1520 A=X
1530 B=Y
1540 FOR I=1 TO LEN(W$)
1550 IF MID$(W$,I,1)<>D$(A,B) THEN I=LEN(W$):
     MATCH=0:GOTO 1600
1560 A=A+U
1570 B=B+V
1580 IF A<1 OR A>15 THEN I=LEN(W$):MATCH=0:
     GOTO 1600
1590 IF B<1 OR B>15 THEN I=LEN(W$):MATCH=0
1600 NEXT I
1610 RETURN
```

```
1620 CH=46080:C=8*ASC("!")
1630 FOR I=0 TO 7
1640 READ D:POKE CH+C+I,D
1650 NEXT I
1655 DATA 0,0,0,12,12,0,0,0
1660 RETURN

1670 PLOT 2,20,"Are you sure that you"
1675 ROW=21:COL=2:GOSUB 2000
1680 INPUT "can find no more words y/n";A$
1685 ROW=20:COL=2:GOSUB 2000:PRINT SPC(35)
1687 ROW=21:COL=2:GOSUB 2000:PRINT SPC(35)
1690 IF A$<>"y" THEN RETURN
1710 CLS

1715 PRINT CHR$(17);CHR$(20)
1720 PRINT "Final score= ";SCORE
1730 INPUT "Another game Y/N";A$
1740 IF A$="Y" THEN RUN
1750 IF A$<>"N" THEN GOTO 1730
1760 STOP
1770 PLOT 1,22,"You got them all"
1780 PRINT CHR$(17);CHR$(20)
1790 GOTO 1730

2000 DOKE #12,48040+ROW*40+COL-1
2010 POKE #268,ROW+1
2020 POKE #269,COL
2030 RETURN
```

# 9
# Treasure Island




Find the hidden treasure before the pirate ship reaches the island. This game has all the ingredients of high adventure. A desert island, peopled by natives both hostile and friendly, gold buried at the spot marked 'X' on the map, quicksands that spell danger to the unlucky treasure-seeker and even Long John Silver's parrot. While you hunt for the booty the pirate ship is heading towards the island and once the pirates land you are certain to be captured. The game is displayed in colour graphics – blue sea and green island – and has sound effects as well.

### How to play

The treasure is buried at the spot marked X on the map that is briefly flashed on to the screen at the start of the game. The path is also indicated. You have to follow the path exactly. If you stray there are

three possible outcomes. If you are lucky Long John Silver's parrot will guide you back to the path – you will see the parrot hovering over the next position on the path, if you are unlucky you will encounter hostile natives and will find yourself back on the path three paces back from where you left it, and if you are unluckier still you will end up in a quicksand. This can be a final fate or you may be rescued by a friendly native. If you need to consult the map in order to follow the path you can type "H". When you do this you will be shown the map – now also indicating the locations of the quicksands – for a short, random length of time. But every time you ask to see the map the pirate ship comes nearer and if the ship arrives before you find the treasure you will be captured. The ship advances in any case once for every five moves you make so you need to be accurate. You can only make your move when the map has disappeared from the screen and the Oric won't allow you to cheat. To move along the path use the right, left and forward arrow keys – you cannot move backwards.

## Subroutine structure

| | |
|---|---|
| 20 | Defines graphics characters |
| 220 | Initialises variables and arrays |
| 310 | Main play loop |
| 510 | Logic for natives' attack |
| 670 | Logic for parrot's help |
| 810 | Prints island with or without map |
| 940 | Prints path |
| 1040 | Moves man |
| 1250 | Prints quicksands on map |
| 1330 | Logic for quicksands |
| 1480 | Constructs island |
| 1600 | Path logic |
| 1680 | Locates treasure |
| 1710 | Constructs quicksands |
| 1800 | Moves and prints pirate ship |
| 2150 | Help routine |
| 2220 | Treasure found routine |
| 2310 | End of game |
| 2500 | Loads character definitions |

## Programming details

This is a very long program with lots of different ingredients. It therefore appears rather complicated whereas in fact it is quite straightforward. One technique to notice is the way the island is constructed at random by subroutine 1480. There the use of SGN function results in the contour of the island first going out and then coming in at random, giving an island-like shape.

## Scope for alteration

If you want to alter the length of time the map is displayed for when you ask to see it you can change the value of the WAIT in line 2170. You might also want to alter the amount that the pirate ship moves at each step by setting a different value for 'R' in line 1840. Currently it's set randomly to a value between 1 and 3.

## Program

```
 10 REM Treasure Island

 20 CH=46080
 30 C=8*ASC("!")
 40 GOSUB 2500
 50 DATA 32,54,62,28,30,39,32,0
 60 C=8*ASC("@")
 70 GOSUB 2500
 80 DATA 1,13,45,63,45,13,18,18
 90 C=8*ASC("#")
100 GOSUB 2500
110 DATA 63,30,45,51,51,45,30,63
120 C=8*ASC("$")
130 GOSUB 2500
140 DATA 8,63,63,63,30,30,12,12
150 C=8*ASC("%")
160 GOSUB 2500
170 DATA 12,28,30,31,63,62,28,24
180 C=8*ASC("^")
190 GOSUB 2500
200 DATA 12,12,30,30,12,18,51,51
```

```
220 F=0
230 XS=1
240 YS=1
250 MES=0
260 DIM V(10)
270 DIM U(10)
280 DIM L(20)
290 DIM R(20)
300 DIM X(20)

310 GOSUB 1480
320 XM=X(T+1)
330 YM=T+1
340 RP=0:GOSUB 810
350 WAIT 300+RND(1)*100
360 RP=1:GOSUB 810
370 GOSUB 1160
380 GOSUB 1040
390 IF XM=XT AND YM=YT THEN GOTO 2220
400 F=F+1
410 IF INT(F/5)=F/5 THEN GOSUB 1800
420 IF X(YM)=XM AND INT(F/5)=F/5 THEN GOTO 360
430 IF X(YM)=XM THEN GOTO 380
440 REM off path
450 PING:GOSUB 1800
460 FOR Q=1 TO 10
470 IF V(Q)=XM AND U(Q)=YM THEN GOSUB 1330:Q=11
480 NEXT Q
500 IF RND(1)<=.4 THEN GOTO 670


510 REM natives
520 RP=1:GOSUB 810
530 PLOT 1,25,"HOSTILE NATIVES AHEAD"
540 FOR N=1 TO 3
550 R=INT(RND(1)*3)+1
560 IF YM+R>=B THEN R=0
590 PLOT XM,YM+R,"@"
600 NEXT N
610 YM=YM-3
620 IF YM<T+1 THEN YM=T+1
630 XM=X(YM)
640 PLOT XM,YM,M$
650 MES=1
660 GOTO 380
```

```
 670 REM parrot
 680 RP=1:GOSUB 810
 690 GOSUB 1160
 720 PLOT 1,25,"FOLLOW LONG JOHN SILVERS PARROT"
 730 YJ=YM+1
 750 XJ=X(YJ)
 780 PLOT XJ,YJ,"!"
 790 MES=1
 800 GOTO 380

 810 REM print island
 820 CLS
 830 PAPER 4
 850 FOR X=L(T)-1TO R(T)+1
 860 PLOT X,T,18
 870 NEXT X
 875 PLOT X,T,20
 880 FOR Y=T TO B
 890 PLOT L(Y)-1,Y,18:PLOT R(Y)+1,Y,20
 900 NEXT Y
 910 FOR X=L(Y-1)-1 TO R(Y-1)+1
 920 PLOT X,Y,18
 930 NEXT X
 933 PLOT X,Y,20
 935 IF RP=1 THEN RETURN

 940 REM print path
 950 FOR Y=T+1 TO P
 960 PLOT X(Y),Y,"*"
 970 NEXT Y
 980 PLOT 1,24,CHR$(7)+"X marks the spot"
 990 PLOT X(P),P,"X"
1000 PLOT XM,YM,"^"
1010 GOSUB 1250
1020 RETURN
```

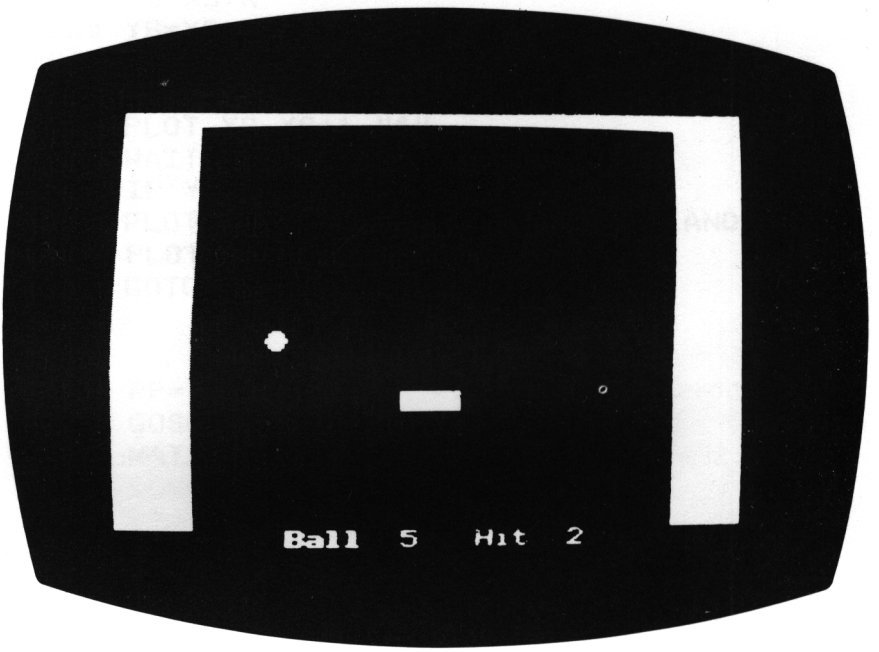
```
1040 REM move man
1045 PLOT XM,YM,"^"
1050 PING
1060 GET A$
1100 PLOT XM,YM," "
1110 IF A$="H" THEN GOSUB 2150:RETURN
1120 IF ASC(A$)=8 THEN XM=XM-1:GOTO 1150
1130 IF ASC(A$)=9 THEN XM=XM+1:GOTO 1150
1140 IF ASC(A$)<>10 THEN GOTO 1050
1150 YM=YM+1
1160 REM plot man
1180 PLOT XM,YM,"^"
1190 IF MES=0 THEN RETURN
1200 FOR I=0 TO 31
1210 PLOT I,23," "
1215 PLOT I,25," "
1220 NEXT I
1230 MES=0:RETURN

1250 REM print quicksands
1270 FOR Q=1 TO 10
1280 PLOT V(Q),U(Q),"%"
1290 NEXT Q
1300 XS=XS+1
1310 YS=YS+1
1320 RETURN

1330 REM quicksand
1340 RP=1:GOSUB 810
1370 PLOT XM,YM,M$
1390 PLOT XM+1,YM+1,"AAARGH"
1410 FOR I=50 TO 10 STEP -4
1420 SOUND 1,I,15:PLAY 1,0,0,0
1425 WAIT 5
1430 NEXT I
1440 PLAY 0,0,0,0
1445 WAIT 100
1450 PLOT 1,20,"IN THE QUICKSAND"
1460 IF RND(1)>.5 THEN PLOT 1,23,"A native
     pulled you out":WAIT 500:RETURN
1470 GOTO 2310
```

```
1480 L=INT(RND(1)*4)+12
1490 T=INT(RND(1)*4)+2
1500 W=15+INT(RND(1)*4)
1510 B=INT(RND(1)*3)+17
1520 FOR Y=T TO B
1530 L(Y)=L
1540 R(Y)=L+W
1550 L=L-INT(SGN(10-Y)*(RND(1)*2))
1560 W=W+INT(SGN(10-Y)*(RND(1)*2))
1570 IF L(Y)<2 THEN L(Y)=2
1580 IF R(Y)>38 THEN R(Y)=38
1590 NEXT Y

1600 X(T+1)=L(T+1)+INT(RND(1)*8)
1610 K=T+2
1620 FOR P=K TO B-2-RND(1)*3
1630 X(P)=X(P-1)+INT(RND(1)*3)-1
1640 IF X(P)>=R(P) THEN X(P)=R(P)-1
1650 IF X(P)<=L(P) THEN X(P)=L(P)+1
1660 NEXT P
1670 P=P-1

1680 XT=X(P)
1690 YT=P

1710 FOR Q=1 TO 10
1720 D=INT(RND(1)*(P-T-2)+T+2)
1730 U(Q)=D
1740 V(Q)=X(D)+SGN(RND(1)-.5)
1750 IF V(Q)<=L(D) THEN V(Q)=V(Q)+3
1760 IF V(Q)>=R(D) THEN V(Q)=V(Q)-3
1770 NEXT Q
1780 RETURN
```

```
1800 REM pirate ship
1810 CLS
1820 INK 7
1830 PAPER 4
1840 R=RND(1)*3
1850 XS=XS+R
1860 YS=YS+R
1870 PLOT 18,18,"%"
1880 PLOT XS,YS,"#"
1890 PLOT XS,YS+1,"$"
1900 WAIT 300
1910 IF YS<18 THEN RETURN
1920 PLOT 0,22,"THE PIRATES HAVE LANDED"
1930 PLOT 0,23,"YOU ARE CAPTURED"
1940 GOTO 2310


2150 RP=0:GOSUB 810
2160 GOSUB 1250
2170 WAIT 100+RND(1)*500
2180 GOSUB 1800
2190 RP=1:GOSUB 810
2200 GOSUB 1160
2210 RETURN

2220 FOR C=0 TO 7
2230 PAPER C
2240 CLS
2250 MUSIC 1,4,C+3,15:PLAY 1,0,0,0
2260 WAIT 50
2270 NEXT C
2275 PAPER 3:INK 0
2280 PLAY 0,0,0,0
2290 PLOT 7,10,"YOU FOUND THE GOLD"
2300 WAIT 100

2310 INPUT "Another game Y/N";A$
2320 IF A$="Y" THEN RUN
2330 PAPER 7
2340 INK 0
2350 CLS
2360 STOP
```

```
2500 FOR I=0 TO 7
2510 READ D
2520 POKE CH+C+I,D
2530 NEXT I
2540 RETURN
```

# 10
# Rainbow Squash




This is a colourful version of the popular computer squash game which is also enhanced by the addition of sound – a cheery beep every time the ball bounces either on the sides of the court or against the bat and a dismal tone every time a ball goes out of play. Its other feature is that as you improve in skill the game gets more difficult and if you then start to get worse it gets easier. This means that your Oric will always give you a challenge that is suited to your ability – which makes it the perfect partner.

## How to play

At the start of the game the bat is at the bottom of the screen and in the centre. You control the left and right movement of the bat by pressing the appropriate arrow keys. Every time you make two hits in succession the position of the bat changes – it moves nearer to the

top of the screen – which makes returning the ball more difficult. If you then miss a shot the bat will move back one position, making it easier. You score a point for every hit and you will be served a total of 10 balls. Information about the number of balls played and hits scored is displayed on the screen continuously.

## Typing tips

Notice that there are single spaces at either side of the three exclamation marks in line 300. They are important as they serve to blank out the old positions of the bat. There are three spaces between the quotes marks in lines 390 and 870 but only one space between the quotes in lines 410 and 660.

## Subroutine structure

| | |
|---|---|
| 20 | Sets up screen and colours and initialise variables |
| 70 | Defines graphics characters |
| 180 | Sets up initial screen display |
| 200 | Main play loop |
| 460 | Draws court |
| 630 | Bounce routine |
| 770 | Moves bat up screen |
| 890 | End of game – prints final score, offers another game and re-runs or restores screen display |
| 1010 | Moves bat to left and right |

## Programming details

The different coloured areas of the court are created by POKEing background attribute codes in the protected column on the far left of the screen. This is carried out by lines 505, 540 and 595.

## Program

```
10 REM Squash
```

```
 20 PRINT CHR$(17);CHR$(6)
 30 HT=0:H=0
 40 D=19
 50 BALL=0
 60 C=2

 70 CH=46080
 80 FOR Q=1 TO 2
 90 READ C
100 FOR I=0 TO 7
110 READ D
120 POKE CH+C*8+I,D
130 NEXT I
140 NEXT Q
150 DATA 33,63,63,63,63,63,63,63,63
160 DATA 64,12,30,30,63,63,30,30,12

180 GOSUB 460
190 X=10

200 BALL=BALL+1
210 IF BALL>10 THEN GOTO 890
220 A=10+INT(RND(1)*7)
230 B=1
240 V=1
250 W=1
260 Y=D
280 M$="Ball "+STR$(BALL):PLOT 10,21,M$
290 GOSUB 1010
300 PLOT X,Y," !!! "
320 M$="Hit "+STR$(HT):PLOT 20,21,M$
340 GOSUB 630
360 IF B+W<>Y THEN Y=D:GOTO 290
370 ZAP
390 PLOT X+1,Y,"   "
400 GOSUB 630
410 PLOT A,B," "
420 IF D<19 THEN D=D+1
430 H=0
440 GOTO 200
```

```
460 CLS
470 PAPER 3
480 FOR I=1 TO 34
490 PRINT "!";
500 NEXT I
505 POKE 48040,17
510 REM COLOR COURT
520 FOR I=1 TO 7
530 PLOT 1,I,"!!!!!":PLOT 31,I,"!!!!!"
540 POKE 48000+(I+1)*40,17
550 NEXT I
580 FOR I=7 TO 20
590 PLOT 1,I,"!!!!!":PLOT 31,I,"!!!!!"
595 IF I<15 THEN POKE 48000+(I+1)*40,20
600 NEXT I
610 RETURN

630 REM BOUNCE
660 PLOT A,B," "
670 A=A+V
680 B=B+W
690 IF A=30 OR A=5 THEN V=-V:PING
700 IF B=1 THEN W=-W:PING
710 IF B+W=Y THEN GOTO 770
750 PLOT A,B,"@"
760 RETURN

770 R=A-X
780 IF R<1 OR R>3 THEN GOTO 750
790 W=-W
800 PING
810 H=H+1
820 HT=HT+1
830 IF H<>1 THEN GOTO 720
840 H=0
850 D=D-1
870 PLOT X+1,Y,"   "
880 GOTO 750
```

```
 890 WAIT 100
 900 CLS
 910 M$="You scored "+STR$(HT)
 920 PLOT 10,10,M$
 930 PLOT 10,15,"Another game y/n "
 940 GET A$
 950 PRINT CHR$(17);CHR$(6)
 960 IF A$="Y" THEN RUN
 970 IF A$<>"N" THEN GOTO 930
 980 CLS:PAPER 7
 990 END

1010 K$=KEY$
1020 IF K$="" THEN RETURN
1030 IF ASC(K$)=9 AND X<26 THEN X=X+1
1040 IF ASC(K$)=8 AND X>5 THEN X=X-1
1050 RETURN
```

# 11
# Commando Jump


This game is a real test of your reaction time and dexterity and is quite compulsive to play. A bright red wall of varying height appears with a little man figure beside it. A countdown "Ready, Steady, Go" is flashed up on the left of the screen and on the word "GO" the man has to jump as high as possible and then scramble up the remainder of the wall in the alotted time – approximately five seconds. Your success in this game depends entirely on your quick wits and nimble fingers.

### How to play

On the word "GO", and no sooner, press any key to make the man jump. The height of the initial jump depends entirely on the delay between the signal appearing and your key press. The quicker you react, the higher the man will jump. The time left to scale the wall is

displayed on the screen and while the rest of your five seconds tick away you must keep on pressing keys to get the man over the wall. The man will climb one brick higher for every four key presses but he will not respond if you simply keep pressing the same key. In other words you have to press different keys alternately – the easiest way is to press two neighbouring keys with two fingers – and the more rapidly you press the more quickly he will climb. If the man is not over within the time limit he will slither back down the wall and you have another try. In all you are given ten attempts. Even if you are very slow off the mark, do press a key – until you do so you cannot move on to the next try. If you hit a key just before the "GO" signal, the computer will accuse you of cheating and you will lose that turn.

## Typing tips

There are six spaces between the quotes in line 70, five in line 100, seven in line 130, six in line 280 and fourteen in line 780. These are needed to blank out messages displayed on the screen. Similarly there are two blanks in line 600 and one each in lines 630, 670, 710, 820 and 880. Notice the null strings in lines 65, 140, 180, 620 and 1165.

## Subroutine structure

| | |
|---|---|
| 20 | Sets up mode and calls initialisation routine |
| 40 | Play loop |
| 60 | Countdown |
| 250 | Cheat routine |
| 330 | Prints well |
| 450 | Jump logic |
| 570 | Scramble over remainder of wall routine |
| 690 | Man falls back down wall |
| 810 | Prints man over wall |
| 970 | Defines graphics characters |
| 1100 | End of game and messages |
| 1230 | Removed attribute codes from M$ |

## Programming details

This is a fairly straightforward application of low resolution dynamic graphics. The wall and the man are both user-defined graphics. The colour of the wall is set by PLOTting the attribute code for red to the left of each brick and the attribute code for black to the right. The reaction time and the count down are produced by using the variable T as a *loop counter*. The Oric has two real time clocks but they both run too fast for this application.

## Scope for alteration

You can make this game easier or more difficult by altering the number of key presses needed to scale one brick higher set in line 640. The value is currently .25 which means that it takes four key presses for the man to scale one interval. Decreasing the value will make the game harder and increasing it will make it easier.

## Program

```
10 REM Commando Jump

20 PRINT CHR$(17);CHR$(6)
30 GOSUB 970

40 GOSUB 330
50 GOSUB 1100
```

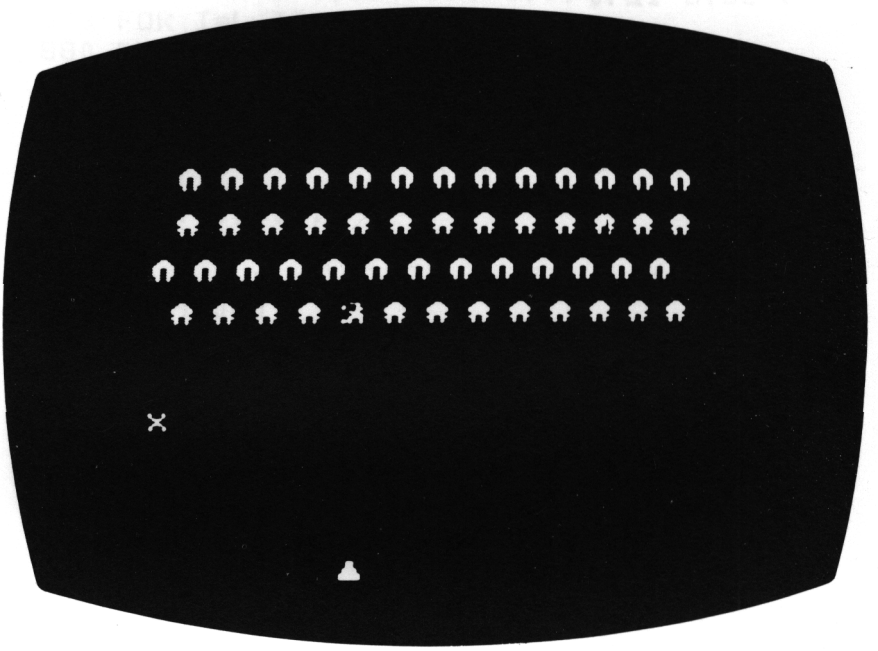
```
 60 PAPER 7
 65 IF KEY$<>"" THEN GOTO 65
 70 PLOT 1,20,"       "
 80 PLOT 2,20,"Ready"
 90 WAIT 100+RND(1)*100
100 PLOT 2,20,"      "
110 PLOT 1,20,CHR$(12)+"Steady"
120 WAIT 100+RND(1)*100
130 PLOT 1,20,"        "
140 IF KEY$<>"" THEN GOTO 250
150 PING
160 PLOT 1,20,"GO"
170 T=0
180 IF KEY$="" THEN T=T+1:GOTO 180
190 T=T/50
200 RETURN


250 PLOT 2,10,"Cheat"
260 SHOOT
270 WAIT 100
280 PLOT 2,10,"      "
290 T=5
300 RETURN

330 REM PLOT WALL
340 CLS
350 JUMP=1
360 H=10+INT(RND(1)*6)
370 FOR I=18 TO 19-H STEP -1
390 PLOT 9,I,CHR$(1)
400 PLOT 10,I,"!"+CHR$(0)
410 NEXT I
420 PLOT 9,18-H,CHR$(1)
430 PLOT 10,18-H,"@"+CHR$(0)
```

```
450 M$="jump "+STR$(JUMP)
460 GOSUB 1230:PLOT 20,2,M$
470 PLOT 13,18,"#"
480 GOSUB 60
490 SOUND 1,18,15:PLAY 1,0,0,0
500 IF T>=5 THEN GOTO 740
510 FOR I=18 TO 18-H+INT(T*28) STEP -1
520 PLOT 13,I," "
530 PLOT 13,I-1,"#"
540 SOUND 1,I*4,15:PLAY 1,0,0,0
550 WAIT 2
555 PLAY 0,0,0,0
560 NEXT I

570 J=I:L=INT(I)
580 T=T+.05
590 IF T>5 THEN GOTO 690
600 M$="time left "+STR$(INT((5-T)*10)/10)+"  "
610 PLOT 5,21,M$
615 K$=KEY$
620 IF K$="" THEN GOTO 580
625 IF K$=J$ THEN GOTO 615
630 PLOT 13,INT(L)," "
635 J$=K$
640 J=J-.25
650 L=INT(J)
660 PLOT 13,L,"#"
670 IF L<=17-H THEN PLOT 13,L+1," ":GOTO 810
680 GOTO 580

690 REM FALL DOWN
700 FOR I=L TO 18
710 PLOT 13,I-1," "
720 PLOT 13,I,"#"
730 SOUND 1,I*4,15:PLAY 1,0,0,0:WAIT 5
735 NEXT I
740 PLAY 0,0,0,0
750 WAIT 10
760 PLAY 0,0,0,0
770 JUMP=JUMP+1
780 PLOT 5,21,"                    "
790 IF JUMP<=10 THEN GOTO 450
800 RETURN
```

```
 810 FOR I=13 TO 5 STEP -1
 820 PLOT I+1,L," "
 830 PLOT I,L,"#"
 840 WAIT 30
 850 NEXT I
 860 SOUND 1,L-1,15:PLAY 1,0,0,0
 870 FOR I=L TO 18
 880 PLOT 5,I-1," "
 890 PLOT 5,I,"#"
 900 SOUND 1,I*4,15:PLAY 1,0,0,0
 910 WAIT 10
 920 NEXT I
 930 PLAY 0,0,0,0
 940 RETURN

 970 CH=46080
 980 FOR Q=1 TO 3
 990 READ C
1000 FOR I=0 TO 7
1010 READ D
1020 POKE CH+C*8+I,D
1030 NEXT I
1040 NEXT Q
1050 DATA 33,0,63,63,63,63,63,63,63
1060 DATA 64,63,63,63,63,63,63,63,63
1070 DATA 35,12,12,63,30,30,18,18,18
1090 RETURN

1100 IF JUMP<=10 THEN GOTO 1130
1110 PLOT 25,19,"You FAILED "
1120 GOTO 1170
1130 M$="You took "+STR$(JUMP)+" jump"
1135 IF JUMP>1 THEN M$=M$+"s"
1140 GOSUB 1230:PLOT 20,5,M$
1160 PLOT 20,6,"to clear the wall"
1165 IF KEY$<>"" THEN GOTO 1165
1170 PLOT 20,8,"Another game Y/N"
1180 GET A$
1190 PRINT CHR$(17);CHR$(6)
1200 IF A$="Y" THEN RUN
1210 PAPER 7:CLS
1220 END
```

```
1230 X=1
1240 IF X>LEN(M$) THEN RETURN
1250 IF ASC(MID$(M$,X,1))>31 THEN X=X+1:
     GOTO 1240
1260 M$=LEFT$(M$,X-1)+MID$(M$,X+1)
1270 GOTO 1240
```

# 12
# Alien Invaders



Invaders from a hostile planet are heading towards earth in two types of spaceships and you and your Oric have to defend civilisation as we know it. Your task is daunting – you have to ensure that none of the aliens get within firing range of earth. The point of no return is marked with an "X" on the left of the screen. Once any of the advancing ships pass this point the game is over – you will have lost. Your only hope is to wipe out the aliens with your missiles. Every missile that hits its target increases your chance of saving the world.

If you play this game on a colour TV you'll see that the ships are red, green and blue.

## How to play

You can move your missile launcher to left and right using the

appropriate arrow keys. Press the 'up arrow' key to fire a missile. You score points for every alien you hit, the ones further away from you counting for more points than the nearer ones. The game is over when you have destroyed all the invaders or when the nearest remaining ones reach the point marked by the X.

## Typing tips

Notice the spacing of the graphics characters in lines 500 and 510. There is a space after the final character in line 500 and a space before the first character in line 510. The effect of this is to ensure that the rows of advancing ships are *staggered* with the ships in the second and fourth rows appearing in the spaces between those in the first and third rows. There are 26 spaces in the E$ string (line 540). It is important that you count these carefully as they are required to test that alien ships have been hit.

## Subroutine structure

| | |
|---|---|
| 20 | Defines graphics characters |
| 400 | Initialises variables and sets background colour to yellow |
| 500 | Initialises strings |
| 1000 | Plots aliens |
| 1100 | Plots and moves missile launcher |
| 1400 | Shoots missile |
| 1500 | Fires missile and test for hit |
| 7000 | Alien hit routine |
| 8000 | Main play loop |
| 8500 | End of game |

## Programming details

The alien ships are stored in strings. The front row (line 500) consists of 13 user-defined graphics characters (@) with blanks after each of them. The second row (line 510) consists of 13 other user-defined characters (%) arranged, by the simple device of printing a blank before each graphics character, so that the ships are staggered in relation to those in the first row. The third row (line 520) repeats the first and the back row (line 530) repeats the second. Line 540 sets up

a string of blank spaces equivalent to the length of each of the rows of invaders. This is used in lines 8160 to 8180 to detect whether any of the strings still contain aliens when they reach the critical screen location marked by the X.

Line 7000 detects when an alien invader is hit. When this happens its position in the string is replaced by a blank space. This manipulation is carried out in line 7010 which divides the string at the appropriate point, inserts a space in place of the destroyed ship and then re-joins the two halves of the string.

## Scope for improvement

You might like to add a routine to make the aliens shoot back at random so that the missile launcher faced the added problem of dodging enemy fire. If you want to make the game easier by lengthening the time before the alien ships move forward a row, increase the value to the right of the ">" sign in line 8010.

## Program

```
10 REM Invaders
```

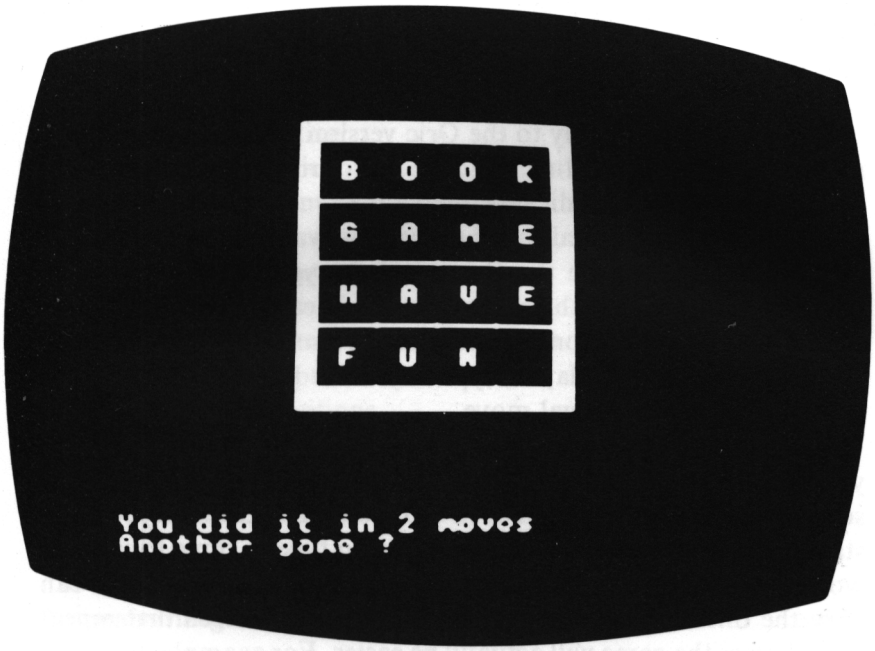
```
 20 CHBAS=46080
 30 POKE CHBAS+(ASC("@")*8+0),12
 40 POKE CHBAS+(ASC("@")*8+1),30
 50 POKE CHBAS+(ASC("@")*8+2),63
 60 POKE CHBAS+(ASC("@")*8+3),51
 70 POKE CHBAS+(ASC("@")*8+4),51
 80 POKE CHBAS+(ASC("@")*8+5),51
 90 POKE CHBAS+(ASC("@")*8+6),18
100 POKE CHBAS+(ASC("@")*8+7),18
110 POKE CHBAS+(ASC("%")*8+0),12
120 POKE CHBAS+(ASC("%")*8+1),30
130 POKE CHBAS+(ASC("%")*8+2),63
140 POKE CHBAS+(ASC("%")*8+3),63
150 POKE CHBAS+(ASC("%")*8+4),30
160 POKE CHBAS+(ASC("%")*8+5),51
170 POKE CHBAS+(ASC("%")*8+6),18
180 POKE CHBAS+(ASC("%")*8+7),18
190 POKE CHBAS+(ASC("#")*8+0),12
200 POKE CHBAS+(ASC("#")*8+1),12
210 POKE CHBAS+(ASC("#")*8+2),12
220 POKE CHBAS+(ASC("#")*8+3),30
230 POKE CHBAS+(ASC("#")*8+4),30
240 POKE CHBAS+(ASC("#")*8+5),63
250 POKE CHBAS+(ASC("#")*8+6),63
260 POKE CHBAS+(ASC("#")*8+7),63
270 POKE CHBAS+(ASC("$")*8+0),10
280 POKE CHBAS+(ASC("$")*8+1),34
290 POKE CHBAS+(ASC("$")*8+2),35
300 POKE CHBAS+(ASC("$")*8+3),10
310 POKE CHBAS+(ASC("$")*8+4),7
320 POKE CHBAS+(ASC("$")*8+5),13
330 POKE CHBAS+(ASC("$")*8+6),41
340 POKE CHBAS+(ASC("$")*8+7),41

400 Y=1
410 XL=10
420 YM=0
430 T=0
440 S=0
450 PAPER 3

500 A$="@ @ @ @ @ @ @ @ @ @ @ @ @ "
510 B$=" % % % % % % % % % % % % %"
520 C$=A$
530 D$=B$
540 E$="                                "
```

```
1000 CLS:PRINT CHR$(17);CHR$(6)
1020 PLOT 1,Y,A$
1030 PLOT 1,Y+4,C$
1050 PLOT 1,Y+2,B$
1060 PLOT 1,Y+6,D$
1070 GOTO 8000

1100 REM MOVE LAUNCHER
1110 T=T+1
1120 PLOT XL,21,"#"
1130 L$=KEY$
1140 IF L$="" THEN RETURN
1150 PLOT XL,21," "
1160 IFASC(L$)=8 AND XL>1 THEN XL=XL-1
1180 IF ASC(L$)=9 AND XL<31 THEN XL=XL+1
1190 PLOT XL,21,"#"

1400 REM SHOOT MISSILE
1410 IF ASC(L$)<>11 THEN RETURN
1415 ZAP
1420 FOR M=19 TO Y+6 STEP -1
1430 PLOT XL,M,","
1440 PLOT XL,M+1," "
1450 NEXT M
1460 PLOT XL,M+1," "
```

```
1500 F=0
1510 Q$=D$:R=6
1520 GOSUB 7000
1530 D$=Q$
1540 IF F=1 THEN GOTO 1750
1550 PLOT XL,Y+5,".":PLOT XL,Y+5," "
1560 PLOT XL,Y+4,".":PLOT XL,Y+4," "
1570 Q$=C$:R=4
1580 GOSUB 7000
1590 C$=Q$
1600 IF F=1 THEN GOTO 1750
1610 PLOT XL,Y+3,".":PLOT XL,Y+3," "
1620 PLOT XL,Y+2,".":PLOT XL,Y+2," "
1630 Q$=B$:R=2
1650 GOSUB 7000
1660 B$=Q$
1670 IF F=1 THEN GOTO 1750
1680 PLOT XL,Y+1,".":PLOT XL,Y+1," "
1690 PLOT XL,Y,".":PLOT XL,Y," "
1700 Q$=A$:R=0
1710 GOSUB 7000
1720 A$=Q$
1750 IF A$=E$ AND B$=E$ AND C$=E$ AND D$=E$
     THEN GOTO 9000
1760 IF Q$=E$ THEN Y=Y+2:PLOT 1,Y-2,E$
1770 GOTO 1100


7000 IF MID$(Q$,XL,1)=" " THEN RETURN
7010 Q$=LEFT$(Q$,XL-1)+" "+MID$(Q$,XL+1)
7020 F=1
7500 S=S+10-Y
7510 Q$=MID$(Q$,2)+LEFT$(Q$,1)
7560 PLOT XL,Y+R,"$"
7570 EXPLODE
7590 M$=STR$(S)
7595 IF ASC(LEFT$(M$,1))<32 THEN
     M$=RIGHT$(M$,LEN(M$)-1)
7596 PLOT 10,25,"SCORE ="+M$
7600 T=T-1
7610 RETURN
```

```
8000 PLOT 1,14,"X"
8010 IF T>100+INT(RND(1)*90) THEN Y=Y+2:T=0:
     PLOT 1,Y-2,E$
8020 A$=MID$(A$,2)+LEFT$(A$,1)
8040 PLOT 0,Y,CHR$(1)+A$
8050 GOSUB 1100
8060 B$=MID$(B$,2)+LEFT$(B$,1)
8070 PLOT 0,Y+2,CHR$(4)+B$
8080 GOSUB 1100
8090 C$=MID$(C$,2)+LEFT$(C$,1)
8100 PLOT 0,Y+4,CHR$(2)+C$
8110 GOSUB 1100
8120 D$=MID$(D$,2)+LEFT$(D$,1)
8130 PLOT 0,Y+6,CHR$(4)+D$
8140 GOSUB 1100
8150 IF Y>14 THEN GOTO 8500
8160 IF Y>12 AND B$<>E$ THEN GOTO 8500
8170 IF Y>10 AND C$<>E$ THEN GOTO 8500
8180 IF Y>8 AND D$<>E$ THEN GOTO 8500
8190 T=T+1
8200 GOTO 8000

8500 PRINT "THEY GOT YOU"
8510 GOTO 9020
9000 PRINT "YOU SAVED THE WORLD"
9020 INPUT "ANOTHER GAME Y/N";Z$
9025 PRINT CHR$(17);CHR$(6)
9030 IF Z$="Y" THEN RUN
9040 INK 0
9050 PAPER 7
9060 CLS
9080 END
```

# 13
# Mirror Tile




Although your Oric opens up lots of new possibilities for games, it's good to know that it can also conjure up old favourites. Mirror Tile is a colourful and versatile version of a game that is conventionally played on a small board with the pieces slotted into one another and into their surrounding frame. This construction is vital to the game, the object of which is to re-arrange the pieces to match a given pattern – hence it's title "Mirror Tile".

If you have never played with a tile puzzle, look at the illustration of the game's display. You'll see a four-by-four square of letters and you'll notice that one position is empty. In other words there are fifteen letters and one hole. The hole allows you to move the letters around the board.

## How to play

Imagine for a moment that the board was really made of plastic pieces. You could slide a piece that was either above or below or to either side of the hole into it and the position of the piece you moved would then be empty – that is, it would become the hole. Notice that there are only a limited number of possible moves – two, three or four depending on the position of the hole, and that it is impossible to move on the diagonals.

These same rules apply to the Oric version of the game. You can move any letter that is directly to the left or right of the hole or just above or below it. To indicate your choice you type in the number (1 to 16) of the square containing the piece you want to move. If you try to make a wrong move the Oric won't let you. Instead it will be helpful and number each square for you, in case your mistake was due to typing in the wrong number for your choice. If you want to see these numbers displayed, type any letter key – in fact any key other than one for a legal move.

When you RUN this program the first thing it asks you is whether you wish to input your own set of words. If you reply "N" then the square will fill with the letters A to O. If you prefer to select your own starting arrangement you will be asked to type in three four-letter words, and one three-letter word. Of course, this means you can vary the difficulty of the puzzle. If you choose words that repeat some letters the game will actually be easier. For example, typing in ROOF, CATS, RENT and TENT would give a fairly simple game. The most difficult puzzle is one where every letter is different, e.g. HOME, CART, WING and SKY.

Once you've typed in your words, you'll see them being shuffled – the program will already have asked you how many shuffles it should perform and the more it shuffles the more difficult you will find it. Then it's your turn – to sort them out again into the initial arrangement. Your Oric will count your moves and let you know how many you took at the end of the game.

## Typing tips

When you type in lines containing messages or input statements you will notice that there are often spaces beyond the last letter of the text and before the final double quotes. Do type in these extra spaces as they are needed to blank out any previous input.

## Subroutine structure

| | |
|---|---|
| 30 | Defines arrays |
| 50 | Sets up game |
| 140 | Main play loop |
| 180 | End of game |
| 230 | Checks for end of game |
| 320 | Sets ups default (alphabetic) board |
| 460 | Prints frame |
| 750 | Shuffle routine |
| 900 | Prints number overlay |
| 990 | Blanks out previous messages |
| 1090 | Move logic |
| 1320 | Locates empty space |
| 1410 | Prints title and initial questions |
| 1550 | Performs move |
| 1610 | Asks for input of words |
| 1950 | Updates positions |
| 2010 | Defines graphics characters and sets up screen |
| 2270 | Moves cursor to position |
| 2310 | Removes attribute codes from Q$ |

## Programming details

This program is complicated both because of its length and also because it involves a lot of logic. However, its subroutine structure is very clear and if you follow it through you should be able to see what happens at every step.

Subroutine 2270 is used to move the text cursor to the row and column position stored in the variables 'ROW' and 'COL'. This subroutine allows you to PRINT text or INPUT from any screen position.

## Scope for improvement

Adding to a program that is already as long as this one may seem to be a tall order. However there is actually scope for improvement. A routine that reminded the player of the target arrangement might be a very useful extra.

**Program**

```
 10 REM Mirror Tile

 30 DIM B(4,4)
 40 DIM B$(16),W$(16)

 50 GOSUB 1410
 60 GOSUB 320
 70 IF WR=1 THEN GOSUB 1610
 80 MOV=0
 90 ER=0
100 GOSUB 2010
110 GOSUB 460
120 GOSUB 1320
130 GOSUB 750

140 GOSUB 1090
150 GOSUB 230
160 MOV=MOV+1
170 IF FIN<>0 THEN GOTO 140

180 Q$=STR$(MOV):GOSUB 2310
182 PLOT 1,24,"You did it in "+Q$+" moves"
183 COL=1:ROW=25:GOSUB 2270
190 INPUT "Another game ";A$
195 PRINT CHR$(17)
200 IF A$="Y" THEN RUN
210 CLS
220 STOP

230 FIN=0
240 K=0
250 FOR I=1 TO 4
260 FOR J=1 TO 4
270 K=K+1
280 IF B$(B(I,J))<>W$(K) THEN FIN=1
290 NEXT J
300 NEXT I
310 RETURN
```

```
320 K=0
330 FOR I=1 TO 4
340 FOR J=1 TO 4
350 K=K+1
360 B$(K)=CHR$(64+K)
370 B(I,J)=K
380 NEXT J
390 NEXT I
400 B$(16)=" "
410 FOR I=1 TO 16
420 W$(I)=B$(I)
430 NEXT I
440 RETURN


460 INK 4
470 PAPER 6
480 FOR I=0 TO 3
490 FOR J=0 TO 3
500 PLOT 10+J*3,4+I*3,"   !"
510 NEXT J
520 FOR J=0 TO 3
520 T$=" "+B$(B(I+1,J+1))+"!"
530 PLOT 10+J*3,5+I*3,T$
540 NEXT J
550 FOR J=0 TO 3
560 PLOT 10+J*3,6+I*3,"@@#"
570 NEXT J
580 NEXT I
590 FOR I=0 TO 11
620 PLOT 10+I,3,ASC("%")+128
640 PLOT 10+I,16,ASC("$")+128
650 PLOT 9,4+I,ASC("(")+128
670 PLOT 22,4+I,ASC("^")+128
680 NEXT I
690 PLOT 9,3,ASC("&")+128
700 PLOT 22,16,ASC("-")+128
710 PLOT 22,3,ASC("[")+128
720 PLOT 9,16,ASC("]")+128
740 RETURN
```

```
750 IS=4:JS=4
760 IY=0:JY=0
770 FOR D=1 TO S
780 I=IS:J=JS
790 IF RND(1)>.5 THEN GOTO 830
800 I=IS+INT(RND(1)*2)*2-1
810 IF I>4 OR I<1 THEN I=IS:GOTO 830
820 GOTO 850
830 J=JS+INT(RND(1)*2)*2-1
840 IF J>4 OR J<1 THEN GOTO 780
850 IF I=IY AND J=JY THEN GOTO 780
860 IY=IS:JY=JS
870 GOSUB 1550
880 NEXT D
890 RETURN

900 K=0
905 ER=1
910 FOR I=0 TO 3
920 FOR J=0 TO 3
930 K=K+1
935 Q$=STR$(K):GOSUB 2310
940 PLOT 10+J*3,4+I*3,Q$
950 NEXT J
960 NEXT I
970 RETURN


990 FOR L=0 TO 3
1000 FOR P=0 TO 3
1010 PLOT 10+P*3,4+L*3,"  "
1020 NEXT P
1030 NEXT L
1040 IF ER=0 THEN RETURN
1050 COL=1:ROW=21:GOSUB 2270
1055 PRINT SPC(64)
1060 ER=0
1070 RETURN
```

```
1090 PLOT 1,20,"What is your move          "
1100 COL=14:ROW=20:GOSUB 2270:INPUT M$
1110 IF M$="" THEN GOTO 1090
1120 IF LEN(M$)<2 THEN M$="0"+M$
1130 IF MID$(M$,1,1)<"0" OR MID$(M$,1,1)>"9"
     THEN GOSUB 900:GOTO 1090
1140 IF MID$(M$,2,1)<"0" OR MID$(M$,2,1)>"9"
     THEN GOSUB 900:GOTO 1090
1150 M=VAL(M$)
1160 IF M>0 AND M<17 THEN GOTO 1220
1170 ER=1
1180 PLOT 1,21,"A move must be a number between "
1190 PLOT 1,22,"1 and 16 as shown"
1200 GOSUB 900
1210 GOTO 1090
1220 I=INT((M-1)/4)
1230 J=M-I*4
1240 I=I+1
1250 IF ABS(I-IS)+ABS(J-JS)=1 THEN GOSUB 1550:
     RETURN
1260 PING
1270 ER=1
1280 PLOT 1,21,"You can only move a tile next to"
1290 PLOT 1,22,"the space           "
1300 GOSUB 900
1310 GOTO 1090

1320 K=0
1330 FOR L=1 TO 4
1340 FOR P=1 TO 4
1350 K=K+1
1360 IF B(P,L)=16 THEN IS=L:JS=P:MS=K
1370 NEXT P
1380 NEXT L
1390 RETURN
```

```
1410 TEXT:PAPER 7:INK 0:PRINT CHR$(17):CLS
1420 PLOT 10,5,"M i r r o r  T i l e"
1430 PLOT 5,10,"Do you want to input your"
1440 PLOT 5,11,"own set of words"
1450 ROW=11:COL=15:GOSUB 2270
1460 INPUT A$
1470 IF A$="Y" THEN WR=1:GOTO 1510
1490 IF A$="N" THEN WR=0:GOTO 1510
1500 CLS:GOTO 1420
1510 PLOT 5,15,"How many shuffles "
1520 ROW=15:COL=18:GOSUB 2270:INPUT S
1530 IF S<1 THEN GOTO 1510
1540 RETURN

1550 GOSUB 990
1560 GOSUB 1950
1570 PLOT 11+(J-1)*3,5+(I-1)*3,B$(B(I,J))
1580 PLOT 11+(JS-1)*3,5+(IS-1)*3,B$(B(IS,JS))
1600 RETURN
```

```
1610 CLS
1620 PRINT "Choose 3 four letter words"
1630 PRINT "and 1 three letter word."
1640 INPUT "Type in the first four letter
     word    ";A$
1650 IF LEN(A$)<>4 THEN GOTO 1640
1660 FOR I=1 TO 4
1670 W$(I)=MID$(A$,I,1)
1680 NEXT I
1690 PRINT "First word= ";A$
1700 INPUT "Type in the second four letter
     word    ";A$
1710 IF LEN(A$)<>4 THEN GOTO 1700
1720 FOR I=5 TO 8
1730 W$(I)=MID$(A$,I-4,1)
1740 NEXT I
1750 PRINT "Second word= ";A$
1760 INPUT "Type the third four letter
     word         ";A$
1770 IF LEN(A$)<>4 THEN GOTO 1760
1780 FOR I=9 TO 12
1790 W$(I)=MID$(A$,I-8,1)
1800 NEXT I
1810 PRINT "Third word= ";A$
1820 INPUT "Type in the three letter
     word            ";A$
1830 IF LEN(A$)<>3 THEN GOTO 1820
1840 FOR I=13 TO 15
1850 W$(I)=MID$(A$,I-12,1)
1860 NEXT I
1870 PRINT "Fourth word= ";A$
1890 W$(16)=" "
1900 FOR I=1 TO 16
1910 B$(I)=W$(I)
1920 NEXT I
1930 RETURN


1950 B(IS,JS)=B(I,J)
1960 B(I,J)=16
1970 T=IS:IS=I
1980 I=T:T=J
1990 J=JS:JS=T
2000 RETURN
```

```
2010 CH=46080
2020 FOR I=0 TO 7
2030 READ D1,D2,D3,D4,D5,D6,D7,D8,D9
2040 POKE CH+I+8*ASC("!"),D1
2050 POKE CH+I+8*ASC("#"),D2
2060 POKE CH+I+8*ASC("@"),D3
2070 POKE CH+I+8*ASC("%"),D4
2080 POKE CH+I+8*ASC("^"),D5
2090 POKE CH+I+8*ASC("&"),D6
2100 POKE CH+I+8*ASC("-"),D7
2110 POKE CH+I+8*ASC("["),D8
2120 POKE CH+I+8*ASC("]"),D9
2125 POKE CH+I+8*ASC("$"),63-D4
2126 POKE CH+I+8*ASC("("),63-D5
2130 NEXT I
2140 DATA 1,1,0,0,48,0,48,0,15
2150 DATA 1,1,0,0,48,0,48,0,15
2160 DATA 1,1,0,0,48,0,48,0,15
2170 DATA 1,1,0,0,48,0,48,0,15
2180 DATA 1,1,0,63,48,15,0,48,0
2190 DATA 1,1,0,63,48,15,0,48,0
2200 DATA 1,1,0,63,48,15,0,48,0
2210 DATA 1,63,63,63,48,15,0,48,0
2250 CLS
2260 RETURN

2270 DOKE #12,48040+ROW*40+COL-1
2280 POKE #268,ROW+1
2290 POKE #269,COL/2
2300 RETURN

2310 X=1
2320 IF X>LEN(Q$) THEN RETURN
2330 IF ASC(MID$(Q$,X,1))>31 THEN X=X+1:
     GOTO 2320
2340 Q$=LEFT$(Q$,X-1)+RIGHT$(Q$,LEN(Q$)-1)
2350 GOTO 2320
```

# 14
# Pot Shot


This game provides the ideal type of target practice. However many times you score a direct hit, the magic bird continues to fly on, allowing you to take aim and fire again and again. The elements of this game are simple – a blue sky with a single pink cloud, a bird winging its way from left to right and your rifle sight. The object is straightforward – to line up the sight with the bird and shoot it. But in practice its not that simple. For one thing every time you fire your rifle 'kicks' to one side or the other so you have to re-align your sight for another shot and for also the birds (and your sight too) disappears behind the cloud when they reach it. A total of five birds fly across the sky and there is no limit to the number of hits you can score – your total for each bird and the whole game are displayed at the end. There is a sound every time you fire your rifle and a distinctively different sound when you hit the bird.

## How to play

To hit the target you have to line the cross point of your sight up with the centre of the bird. Use all four arrow keys to move your sight and press the space bar to fire. There are five birds in all and you may hit each one as often as you can.

## Subroutine structure

| | |
|---|---|
| 20 | Sets mode and call initialisation routine |
| 40 | Main play loop |
| 110 | Plots cross |
| 180 | Moves sight, calls firing routine and moves bird |
| 390 | Draws cloud and sets up display |
| 580 | Draws bird |
| 680 | Shoots, tests for hit and recoils sight |
| 740 | Sets up screen, defines graphics characters and envelope |
| 940 | End of game |

## Programming details

The rifle shot sound is produced using the Oric's pre-programmed SHOOT command in line 680. The use of high resolution graphics in this program means that the range and smoothness of both the bird and the rifle sight is better than could be achieved with low-resolution graphics. Both the bird and the cross are drawn using the *invert colour* action of the CURSET and DRAW commands. This changes background points to foreground points the first time a shape is drawn and vice versa the second time the shape is drawn. In this way both the cross and the bird are self blanking – drawing them once makes them appear and drawing them again in the same place makes them disappear. Another interesting point about the Oric's colour handling can be seen in the use of the FILL commands, lines 460 and 480, to change an area of the screen to background points displaying inverted colour. As the background colour is cyan the cloud area produced by FILL shows red. The function POINT is used in line 685 to discover if the target has been hit. It is also interesting to note the way the bird's flight path is calculated and stored in the array B in line 800 for use later in the program.

### Scope for improvement

To make this program even more of a challenge you could add more than one cloud and allow the bird to have more than one path across the sky.

### Program

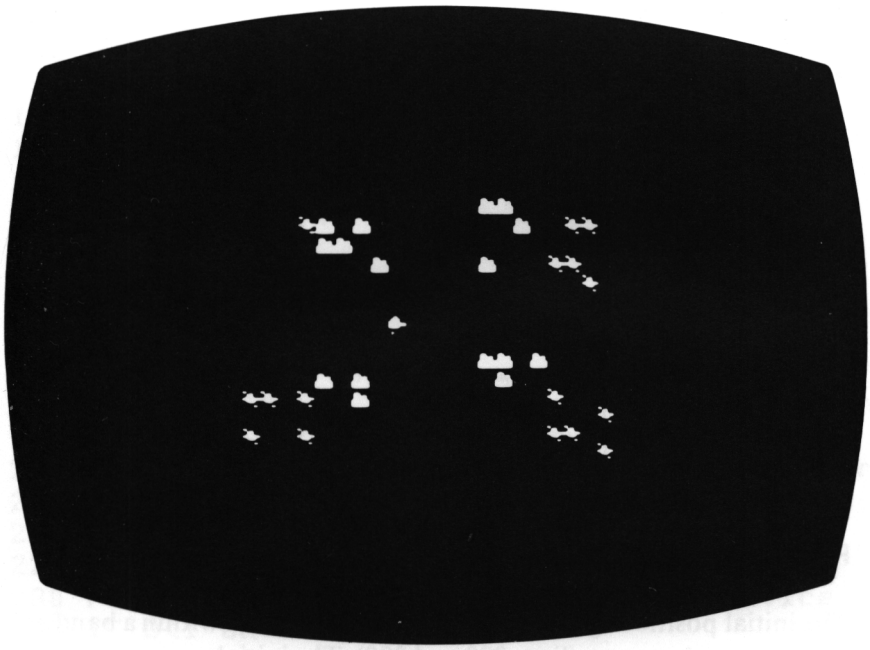
```
 10 REM Pot Shot

 20 PRINT CHR$(17);CHR$(6)
 30 GOSUB 740

 40 FOR G=1 TO 5
 50 GOSUB 390
 60 PRINT "Bird ";G;
 70 GOSUB 180
 80 NEXT G
 90 GOTO 940
100 STOP

110 REM cross
120 CURSET X-6,Y,2
130 DRAW 12,0,2
140 CURSET X,Y-6,2
150 DRAW 0,12,2
160 RETURN
```

```
180 A$=KEY$
190 GOSUB 110
200 J=B(I)
210 GOSUB 580
220 IF A$="" THEN GOTO 270
230 IF ASC(A$)=8 AND X>15 THEN X=X-8
240 IF ASC(A$)=9 AND X<225 THEN X=X+8
250 IF ASC(A$)=10 AND Y<175 THEN Y=Y+8
260 IF ASC(A$)=11 AND Y>15 THEN Y=Y-8
270 I=I+4
280 J=B(I)
290 GOSUB 580
300 IF A$=" " THEN GOSUB 680
310 GOSUB 110
320 IF FIN=1 THEN FIN=0:RETURN
330 WAIT 2
340 GOTO 180


390 REM start
400 HIRES
410 PRINT CHR$(17)
420 PAPER 6:INK 0
430 J=INT(RND(1)*30)+15
440 R=INT(RND(1)*100)+20
450 CURSET R+16,J,0
460 FILL 10,5,128
470 CURSET R+8,J+8,0
480 FILL 10,5,128
500 FIN=0
510 X=INT(RND(1)*30)+30
520 Y=75
530 I=25
540 GOSUB 110
550 J=B(I)
560 GOSUB 580
570 RETURN
```

```
580 CURSET I-8,J,2
590 DRAW 2,-2,2
600 DRAW 2,0,2
610 DRAW 2,+2,2
620 DRAW 4,0,2
630 DRAW 2,-2,2
640 DRAW 2,0,2
650 DRAW 2,+2,2
655 CURSET I-2,J+1,2:DRAW 4,0,2
656 CURSET I-2,J+2,2:DRAW 4,0,2
660 IF I>228 THEN FIN=1
670 RETURN

680 SHOOT
685 P=POINT(X,Y)+POINT(X,Y+2)+POINT(X,Y-2)
690 IF P=0 THEN X=X+10-INT(RND(1)*20):RETURN
700 X=X+10-INT(RND(1)*20)
710 ZAP
720 H(G)=H(G)+1
730 RETURN

740 HIT=0
750 TEXT:CLS
760 PAPER 6:INK 0
770 PLOT 10,10,"P O  T  S  H O T"
780 DIM B(235)
790 FOR I=1 TO 235 STEP 4
800 B(I)=INT(30+(125-I)*(125-I)/200)
810 NEXT I
820 DIM H(5)
830 RETURN
```

```
 940 TEXT:CLS
 950 INK 7:PAPER 0:T=0
 960 FOR I=1 TO 5
 970 M$="Bird "+STR$(I)+" hit  "+STR$(H(I))
 980 PLOT 10,I+5,M$
 990 NEXT I
1000 M$="Total hits= "+STR$(T)
1010 PLOT 10,12,M$
1020 PLOT 10,14,"Another game Y/N
1030 GET A$
1040 PRINT CHR$(17);CHR$(6)
1050 IF A$="Y" THEN RUN
1055 IF A$<>"N" THEN GOTO 1020
1060 CLS
1070 STOP
```

# 15
# Save the Whale



This is a moving graphics game for conservationists! The object of the game is to ensure that the whale survives to swim on in arctic seas. You have to outwit the eskimos who are hunting the whale in their kayaks. If they run into the icebergs they will have to abandon their hunt so you must lure them towards these obstacles by moving the whale in such a way that, in approaching it, the eskimos crash.

## How to play

At the beginning of the game you can select the difficulty level for your turn. Your selection governs the starting positions of the icebergs and so makes the game easier or harder to play. To move the whale, which stands out by being black on a yellow ground, you press any of the arrow keys. You will know when to move as you will hear a signal. If any kayak runs into an iceberg the kayak vanishes, if

the whale runs into an iceberg the iceberg vanishes – this of course reduces his protection so it's not advisable except in extreme circumstances – and if an eskimo reaches the whale he harpoons the whale and kills him. The game is over when all the eskimos have been removed from play or when the whale is dead.

## Subroutine structure

15 Sets up graphics characters and arrays
150 Title frame
280 Prints kayaks
350 Prints icebergs
410 Prints whale
450 Main play loop
520 Checks for game over
540 Move whale routine
680 Move kayaks routine
790 End of game

## Programming details

The initial positions of the kayaks are set at random within a band at the edges of the screen (lines 290 and 300). The initial positions of the icebergs are set in a similar fashion (lines 370 and 380) but account is also taken of the difficulty factor, 'D', input at 230. The SCRN function is used in line 740 to detect whether a kayak has landed on an iceberg – in which case that is the end of the kayak. The alternative method of simply comparing co-ordinates is used in line 750 to detect whether a kayak has landed on the whale – in which case that is the end of the whale (and the game).

## Program

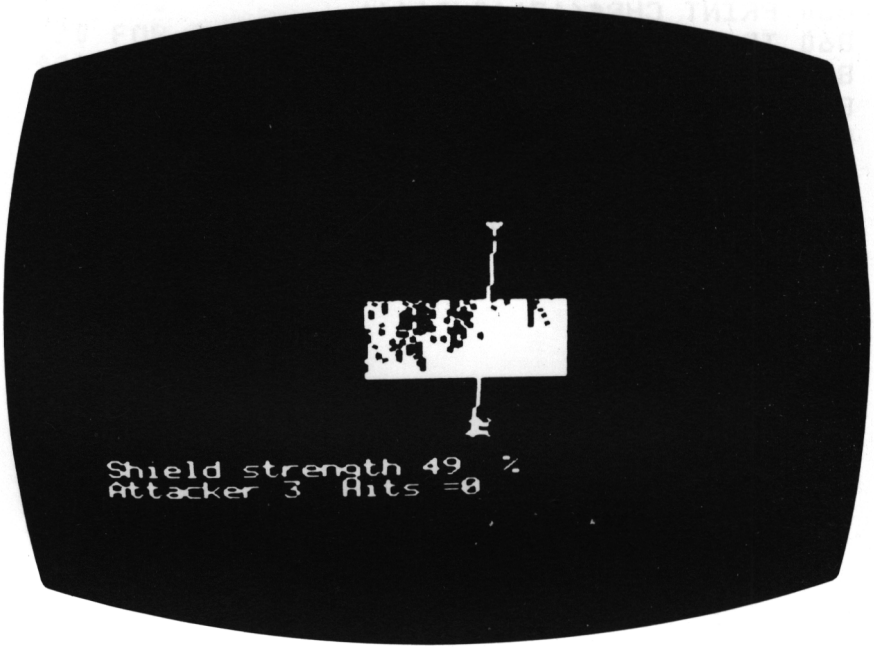
```
10 REM Save the Whale
```

```
 15 DIM X(20),Y(20),U(20),V(20)
 20 PRINT CHR$(17);CHR$(6)
 30 CH=46080
 40 FOR Q=1 TO 3
 50 READ C
 60 FOR I=0 TO 7
 70 READ D
 80 POKE CH+C*8+I,D
 90 NEXT I
100 NEXT Q
110 DATA 33,0,0,24,60,61,63,61,0
120 DATA 64,0,16,58,62,63,63,63,0
130 DATA 35,0,36,44,28,63,31,4,3

150 CLS
155 PAPER 4:INK 7
160 PLOT 2,2,"S A V E  T H E  W H A L E"
170 PLOT 2,5,"In this game, you, the whale !"
180 PLOT 2,7,"must outwit the eskimoes hunting"
190 PLOT 2,8,"you in their kayaks  #"
200 PLOT 2,10,"by luring them onto the icebergs @"
210 PLOT 2,12,"Which difficulty level do you wish to"
220 PLOT 2,13,"play at...."
225 PLOT 13,15,"(1) expert"
230 PLOT 5,17,"(2) intermediate, (3) novice ":GET D
240 IF D<1 OR D>3 THEN GOTO 230
250 CLS
260 PAPER 4

280 FOR C=1 TO 20
290 X=SGN(RND(1)-.5)*(INT(RND(1)*5)+10)+15
300 Y=SGN(RND(1)-.5)*(INT(RND(1)*5)+6)+11
310 PLOT X,Y,"#"
320 X(C)=X
330 Y(C)=Y
340 NEXT C

350 REM ICEBERGS
360 FOR C=1 TO 20
370 U(C)=SGN(RND(1)-.5)*(INT(RND(1)*5)+5-D)+15
380 V(C)=SGN(RND(1)-.5)*(INT(RND(1)*5)+3-D)+9
390 PLOT U(C),V(C),"@"
400 NEXT C
```

```
410 REM PRINT WHALE
420 X=INT(RND(1)*2)+10
430 Y=INT(RND(1)*2)+10
440 PLOT X,Y,ASC("!")+128

450 GOSUB 540
460 F=0
470 FOR C=1 TO 20
480 IF X(C)=0 THEN GOTO 510
490 F=1
500 GOSUB 680
510 NEXT C

520 IF F=0 THEN GOTO 830
530 GOTO 450

540 PING
550 Z=X:V=Y
560 GET A$
570 IF ASC(A$)=8 AND X>1 THEN X=X-1
580 IF ASC(A$)=9 AND X<31 THEN X=X+1
590 IF ASC(A$)=10 AND Y<20 THEN Y=Y+1
600 IF ASC(A$)=11 AND Y>1 THEN Y=Y-1
610 REM BLANK WHALE
620 PLOT Z,V," "
630 REM PRINT WHALE
640 PLOT X,Y,ASC("!")+128
650 RETURN

680 PLOT X(C),Y(C)," "
690 E=0
700 E=SGN(X(C)-X)
710 X(C)=INT(X(C)-E)
720 E=SGN(Y(C)-Y)
730 Y(C)=INT(Y(C)-E)
740 IF SCRN(X(C),Y(C))=64 THEN X(C)=0:ZAP:
    GOTO 780
750 IF X(C)=X AND Y(C)=Y THEN GOTO 790
760 REM PRINT KAYAK
770 PLOT X(C),Y(C),"#"
780 RETURN
```

```
790 REM KILL WHALE
800 PLOT X(C),Y(C)," "
810 PLOT 1,19,"You were killed"
820 GOTO 840
830 PLOT 1,19,"You escaped this time "
840 PLOT 1,20,"Another game Y/N":GET A$
850 PRINT CHR$(17);CHR$(6)
860 IF A$="Y" THEN RUN
865 IF A$<>"N" THEN GOTO 840
870 CLS
880 PAPER 7
890 INK 0
900 STOP
```

# 16
# Mighty Missile




Your weapon can destroy anything – anything that it actually hits. So the only problem in this game is to ensure that the missile finds its target, quickly and accurately. The enemy ships sweep in from the left and the right firing relentlessly. Your missile is only vulnerable if its protective shield is eroded away and then it can be easily blasted in its home base. Otherwise, it is impervious to enemy fire, even outside its base. If it hits an enemy it will explode on contact but if it fails to find its target it will disintegrate as it reaches the upper atmosphere. You can launch ten missiles and there are ten enemy ships. Each ship maintains a stable orbit until you actually take a shot at it – so you can wait in the base while deciding which side to fire from and when to fire – except that with every orbit more of your shield is blasted away by enemy fire and once there is only 50% of it left you will no longer have any protection from the enemies' lasers. This fast moving graphics game is enhanced by sound effects and is quite compulsive to play.

## How to play

In this game it is important to notice how much 'Shield strength' you have left since when the figure displayed drops to 50% you will be vulnerable to attack. Once the shield is so eroded the enemy laser is able to home in and destroy you wherever they hit you, including inside the missile base. The object of the game is to score as many hits as possible so it is worth watching each new enemy skip's orbit at least once or twice before you try to shoot it down. To fire you have to leave your base. Do this by pressing the right or left arrow key. This will take you to a fixed position on the right or left of the screen and launch the missile. Remember to take into account the time it will take for your missile to reach the enemy ship which will continue on its path! At the end of the game your score is displayed and you are given the option of another game.

## Subroutine structure

| | |
|---|---|
| 20 | Set up |
| 70 | Main play loop |
| 310 | End of game |
| 380 | Prints attacker and fire laser |
| 500 | Checks to see if player has activated missile |
| 620 | Calculates shield strength |
| 710 | Moves and fires missile and tests for hit or miss |
| 780 | Explosion graphics and sound |
| 940 | Sets up attack orbit |
| 1070 | Prints shield |
| 1200 | Defines graphics characters and sets up screen display |

## Programming details

As this program has a clear structure and uses separate procedures for most of its major elements, you should find it relatively easy to follow. One interesting point to note is the way in which the path of the attacking ship is calculated in subroutine 940 and stored in a pair of arrays X and Y to be used repeatedly for the various orbits used during the game. A second point of interest is that although high resolution graphics are used for the laser zap and the shield the spaceships and the explosion are both low resolution user-defined graphics characters. The shield is an area of inverse foreground colour produced by the FILL command. As the laser beam works by first drawing a line in the foreground colour and then blanking this out by drawing a line in the background colour, it whittles the shield away by changing parts of it to inverse background colour – red. The strength of the shield is estimated, in subroutine 620, by the number of foreground points left in the shield, using the POINT function. POINT is −1 if the point at the x,y co-ordinate is foreground and 0 if it is background.

## Program

```
10 REM Mighty Missile

20 TEXT:PAPER 6:INK 0:CLS
30 GOSUB 1200
40 GOSUB 940
50 GOSUB 1070
60 GOSUB 620
```

```
 70 FOR A=1 TO 10
 80 DIR=SGN(RND(1)-0.5)
 90 PRINT "Attacker ";A;" Hits =";HIT
110 R=7-INT(RND(1)*14)+1
120 IF R<0 THEN S=5-R:E=36
130 IF R>=0 THEN S=5:E=36-R
140 IF DIR=-1 THEN T=S:S=E:E=T
150 FOR I=S TO E STEP DIR
160 GOSUB 380
170 GOSUB 50
180 IF F=1 THEN GOSUB 710
190 NEXT I
200 CURSET X(I),Y(I+R),3:CHAR ASC("&"),0,0
210 IF F=1 THEN FIN=1
220 IF FIN=2 THEN A=11:GOTO 300
230 GOSUB 620
240 IF F=1 THEN F=0:CURSET MX,MY,3:
    CHAR ASC("!"),0,1:EXPLODE:WAIT 200
250 CURSET MX,MY,3:CHAR ASC("@"),0,0
255 CURSET MX,MY,3:CHAR ASC("!"),0,0
260 MX=120:MY=185
270 CURSET MX,MY,3:CHAR ASC("@"),0,1
280 IF FIN=0 THEN GOTO 150
290 FIN=0
300 NEXT A

310 WAIT 200
320 TEXT:CLS
330 IF FIN=2 THEN PRINT "THEY GOT YOU"
340 PRINT:PRINT "YOU HIT ";HIT
350 PRINT:INPUT "ANOTHER GAME";A$
360 IF A$="Y" THEN RUN
365 STOP
```

```
380 REM ENEMY
390 CURSET X(I),Y(I+R),3:CHAR ASC("&"),0,0
400 CURSET X(I+DIR),Y(I+R+DIR),3:CHAR ASC("&"),0,1
410 IF RND(1)<.5 THEN RETURN
415 IF F=1 THEN GOTO 430
420 IF C<=50 AND ABS(MX-X(I+DIR))<6 THEN FIN=2:
    GOTO 840
430 CURSET X(I+DIR)+3,Y(I+R+DIR)+6,1
440 D=20-INT(RND(1)*40)
450 DRAW D,40,1
460 ZAP
470 CURSET X(I+DIR)+3,Y(I+R+DIR)+6,0
480 DRAW D,40,0
490 RETURN

500 REM FIRE
510 IF F=1 THEN RETURN
520 A$=KEY$
530 IF A$="" THEN RETURN
540 CURSET MX,MY,3:CHAR ASC("@"),0,0
550 IF ASC(A$)=8 THEN  MX=MX-45:GOTO 590
560 IF ASC(A$)=9THEN  MX=MX+45:GOTO 590
570 RETURN
590 CURSET MX,MY,3:CHAR ASC("@"),0,1
600 F=1
610 RETURN

620 REM SHIELD STRENGTH
630 C=0
640 J=145
650 FOR I=91 TO 149
660 C=C-POINT(I,J)
670 NEXT I
680 C=INT(C/59*100)
690 PRINT "Shield strength ";C;" %"
700 RETURN

710 REM GUIDE MISSILE
720 CURSET MX,MY,3:CHAR ASC("@"),0,0
730 MY=MY-6
740 IF MY<10 THEN F=0:FIN=1:GOTO 860
750 CURSET MX,MY,3:CHAR ASC("@"),0,1
760 IF ABS(MX-X(I+DIR))>6 THEN RETURN
770 IF ABS(MY-Y(I+R+DIR))>6 THEN RETURN
```

```
 780 FIN=1
 790 CURSET MX,MY,3:CHAR ASC("@"),0,0
 795 CURSET X(I+DIR),Y(I+R+DIR),3
 820 HIT=HIT+1
 830 F=0
 835 GOTO 860
 840 CURSET X(I+DIR)+3,Y(I+R+DIR)+6,3
 850 DRAW MX-X(I+DIR)-1,MY-Y(I+R+DIR)-6,1
 860 CHAR ASC("!"),0,1
 870 EXPLODE
 880 WAIT 150
 890 IF FIN=2 THEN GOTO 920
 900 CHAR ASC("!"),0,0
 910 CURSET X(I+DIR),Y(I+R+DIR),3:CHAR ASC("&"),0,0
 920 I=E+DIR
 930 RETURN

 940 REM CALCULATE PATH
 950 DIM X(50),Y(50)
 960 X=0:Y=0
 970 N=39
 980 FOR I=1 TO N
 990 X=X+5
1000 Y=120-INT((100-X)*(100-X)/100)
1010 X(I)=X
1020 Y(I)=Y
1030 NEXT I
1040 I=1
1050 HIT=0
1060 RETURN

1070 REM DRAW SHIELD
1080 CURSET 90,140,3
1090 FILL 30,10,255
1100 MX=120:MY=185
1110 CURSET MX,MY,3
1120 CHAR ASC("@"),0,1
1130 F=0
1140 FIN=0
1150 RETURN
```

```
1200 REM DEFINE GRAPHICS
1210 CH=46080
1220 FOR Q=1 TO 3
1230 READ C
1240 FOR I=0 TO 7
1250 READ D
1260 POKE CH+C*8+I,D
1270 NEXT I
1275 NEXT Q
1280 DATA 33,2,33,6,38,48,6,22,33
1290 DATA 64,12,30,12,12,12,30,63,0
1300 DATA 38,21,31,14,4,4,4,0,0
1310 HIRES
1320 PAPER 6
1330 INK 0
1340 CLS
1350 RETURN
```

# 17
# Nine Hole Golf


This is a graphics game that combines both driving and putting and even includes the hazard of bunkers. You play around a nine hole course with two stages at each hole – the fairway and the green. When you RUN it notice how, in the first stage, the golfer makes his swing and how the ball flies through the air.

## How to play

At the start of each hole you are told the distance to the hole – marked on the screen by a flag – and asked to select which club you wish to use. If you've ever played golf, or watched it on TV, you'll know that the lower the number of the club the further it will drive the ball. In other words, select the 1 iron to drive a long way and 8 iron for the really close shots. If you land in a bunker you'll find that you next shot is not as effective as it normally would be. If you

overshoot the green you'll get a new go at the hole and if you drive the ball off the screen you forfeit the hole and move on to the next one. Otherwise, once you get close enough to the hole you'll find yourself on the green. A message will tell you how far you have to putt to the hole and will ask you to select the appropriate club. If you overshoot while putting you will find yourself still at some distance from the hole and will have to carry on putting until your ball drops in to the hole. Your score for each hole is displayed at the end of each hole and a score card for all nine holes is displayed at the end of each round.

## Typing tips

As well as the three user-defined graphics characters, you will also find a capital 'O' used in this program in line 1400. It marks the hole on the putting green.

## Subroutine structure

| | |
|---|---|
| 20 | Defines graphics characters |
| 130 | Initialises variables |
| 170 | Sets up and plays each hold |
| 670 | Reports score for each hole |
| 765 | End of game |
| 800 | Plots balls' flight |
| 1100 | Lost ball routine |
| 1160 | Displays swinging club |
| 1320 | Putting routine |
| 1700 | Moves cursor to position |

## Programming details

An interesting feature of this game is the use of high resolution graphics to make the player appear to swing his club. This is done in subroutine 1160 which draws a line which is the continually shifting radius of a circle. The flight of the ball is also plotted using high resolution graphics. The path that the ball appears to follow (subroutine 800) is a distorted parabola that always carries the ball in the direction of the flag.

## Scope for improvement

You may have noticed that the score card at the end of the game does not total your score nor compare it with any ideal *par* for the course. You might like to add both these features. You will need to play the game a few times to discover what figure to set as the par.

## Program

```
 10 REM Golf

 20 CH=46080
 30 FOR Q=1 TO 3
 40 READ C
 50 FOR I=0 TO 7
 60 READ D
 70 POKE CH+C+I,D
 80 NEXT I
 90 NEXT Q
100 DATA 264,8,12,14,8,8,8,8,62
110 DATA 512,12,30,45,30,12,18,18,33
120 DATA 280,12,30,31,46,62,60,28,24

130 DIM T(9)
140 XH=0:XC=0
150 YH=0:HT=0:YC=0
```

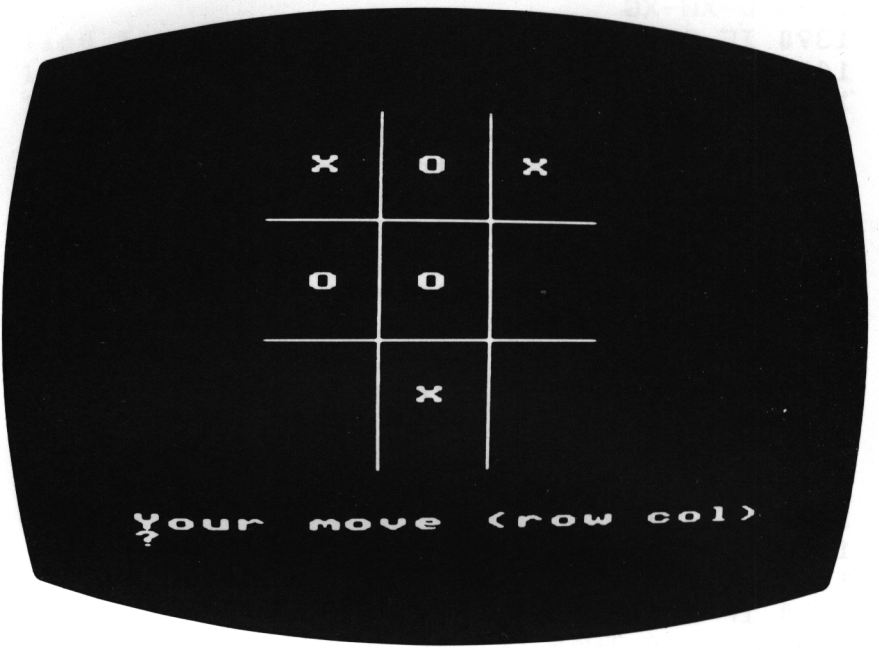
```
170 FOR H=1 TO 9
180 HIRES
190 PAPER 2:INK 7
200 PRINT "Hole Number ";H
220 FOR Z=1 TO 8
230 XB=INT(RND(1)*80)+80
240 YB=INT(RND(1)*40)+50
250 CURSET XB,YB,0
260 CHAR 35,0,1
270 NEXT Z
300 X=INT(RND(1)*20)+20
310 Y=INT(RND(1)*30)+120
320 XT=INT(RND(1)*50)+150
330 YT=INT(RND(1)*10)+20
340 D=SGN(XT-X)*SQR((XT-X)*(XT-X)+(YT-Y)*(YT-Y))
350 CURSET XT,YT,0
360 CHAR 33,0,1
370 CURSET X,Y,0
380 CHAR 64,0,1
400 PRINT "Distance to next hole is ";INT(D)
410 INPUT "Which club (1 to 8)";C
420 IF C<1 OR C>8 THEN GOTO 420
430 C=(9-C)
440 GOSUB 1160
480 GOSUB 800
490 B=1
500 D=SGN(XT-X1)*SQR ((XT-X1)*(XT-X1)+(YT-Y1)*
    (YT-Y1))
510 IF D<-15 THEN PRINT "You overshot-try
    another hole":WAIT 100:GOTO 180
520 IF D<12 THEN PRINT "ON THE GREEN":
    WAIT 200:GOTO 1320
530 CURSET X1,Y1,0
540 CHAR 46,0,0
550 CURSET X,Y,0
560 CHAR 64,0,0
570 X=X1
580 Y=Y1
590 GOTO 350
640 PRINT "You took ";T(H);" strokes"
650 WAIT 200:PRINT CHR$(17)
660 NEXT H
```

```
 670 CLS
 680 PRINT:PRINT
 690 PRINT SPC(15);"This Round"
 700 COL=1:ROW=5:GOSUB 1700
 720 FOR I=1 TO 9
 730 PRINT SPC(10);"Hole ";I;
 740 IF T(I)=-1 THEN PRINT "  lost ball":
     GOTO 760
 750 PRINT " ";T(I);" strokes"
 760 NEXT I

 765 ROW=20:COL=5:GOSUB 1700
 770 INPUT "Another round Y/N";A$
 775 PRINT CHR$(17)
 780 IF A$="Y" THEN RUN
 790 CLS:STOP

 800 VT=C*(1+RND(1)*.01)
 810 HT=0
 820 XH=0
 840 Q=(Y-YT)/(XT-X)
 850 VV=-VT*(SIN(45*PI/180))
 860 XC=X+3
 870 YC=Y+3
 880 VH=VT*(COS(45*PI/180))
 890 HT=HT+VV
 900 YH=Q*XH
 910 VV=VV+1
 920 XH=XH+VH
 930 YH=-Q*XH
 940 IF XH+XC>250 THEN GOTO 1100
 950 IF YH+HT+YC<0 THEN GOTO 1100
 960 IF HT>=0 THEN GOTO 1040
 970 CURSET XH+XC,YH+HT+YC,0
 980 CHAR 46,0,1
1000 WAIT 3
1010 CURSET XH+XC,YH+HT+YC,0
1020 CHAR 46,0,0
1030 GOTO 890
1040 X1=XH+XC
1050 Y1=YH+HT+YC
1070 CURSET X1,Y1,0
1080 CHAR 46,0,1
1090 RETURN
```

```
1100 YH=0:YT=0:YC=0:XH=0:XC=0
1110 PRINT "You've lost your ball "
1120 PING
1130 T(H)=-1
1140 WAIT 300
1150 POP:GOTO 660

1160 T(H)=T(H)+1
1170 XS=X+4
1180 YS=Y+4
1190 FOR S=-5 TO -40 STEP -5
1200 A=S/30*PI
1210 SX=10*SIN(A)
1220 SY=-10*COS(A)
1230 CURSET XS,YS,0
1240 DRAW SX,SY,2
1250 IF S<>-30 THEN WAIT 10
1260 IF S=-30 THEN PING
1270 CURSET XS,YS,0
1280 DRAW SX,SY,2
1300 NEXT S
1310 RETURN
```

```
1320 TEXT:PRINT CHR$(17)
1330 PAPER 2
1340 XG=INT(RND(1)*5)+1
1350 YG=15
1360 XH=INT(RND(1)*15)+10
1370 YH=15
1380 D=XH-XG
1390 IF D<0 THEN D=ABS(D)
1400 PLOT XH,YH,"O"
1410 PLOT XG,YG,"@"
1420 ROW=22:COL=1:GOSUB 1700
1440 PRINT "Distance to hole is ";D;"    "
1450 INPUT "Which club (1 to 8)";C
1460 IF C<1 OR C>8 THEN GOTO 1450
1470 T(H)=T(H)+1
1480 H1=8-C+INT(RND(1)*2)
1485 IF D=1 AND C=8 THEN H1=1
1490 D=D-H1
1500 FOR Z=XG+1 TO XG+H1
1510 PLOT Z,YH,"."
1520 WAIT 10
1530 PLOT Z,YH," "
1540 NEXT Z
1550 PLOT XG,YG," "
1560 XG=XG+H1
1570 IF XG=XH THEN GOTO 640
1580 IF D<0 THEN CLS:XG=XH+D:GOTO 1380
1590 GOTO 1400


1700 DOKE #12,48040+ROW*40+COL-1
1710 POKE #268,ROW+1
1720 POKE #269,COL/2
1730 RETURN
```

# 18
# Noughts and Crosses




Noughts and crosses is a perennial favourite because it is a simple game of strategy. The problem with playing it against a computer is that the computer can be programmed so that the person challenging it can never win. However, this program makes your Oric an opponent who can be beaten. The Oric will make sensible moves but is not infallible so it is worth playing on until you beat it. It's actually a very good way of learning about game-playing strategy.

### How to play

This game is played on a simple three-by-three grid in the traditional way. You have the 'X' and play first. To make your move you have to specify which square to place your mark on. Type in the row number first, then the column number. For example, type

11 to place your nought in the top, lefthand corner. If you type a number in the wrong format (for example 1,1) or a number that does not correspond to a position on the grid, (for example 41) the Oric won't accept it and will beep at you. If you type the number of a position that is already occupied a message to that effect will be displayed. Once you've made your move the computer replies with its 'O' and you make your next move. At the end the Oric will display "I WIN" if it has been successful, "YOU WIN" if you've been successful and "DRAW" if it's stalemate. The board has to be completely filled for the game to be over.

## Typing tips

A capital 'O' is used in this program – look out for it in the CHAR statement in line 950.

## Subroutine structure

| | |
|---|---|
| 20 | Main play loop |
| 120 | Evaluates computers move |
| 630 | Tries each move |
| 780 | Gets player's move |
| 920 | Displays moves |
| 1020 | Sets up screen and plots frame |
| 1220 | End of game |

## Programming details

The method used for the computer to play noughts and crosses is based on an advanced technique from artificial intelligence. The program only looks one move ahead when deciding its move – in other words it does not try to take account of the next move you'll make – which is why it slips up sometimes and allows you to win!

The board is drawn in high resolution graphics and the "X"s and "O"s are placed on the screen using the CHAR command. The messages and input are all accommodated in the two text lines at the bottom of the high res screen.

**Program**

```
 10 REM Noughts and Crosses

 20 GOSUB 1020
 30 GOSUB 780
 40 GOSUB 920
 50 GOSUB 630
 60 IF FIN=1 THEN GOTO 1260
 70 IF FIN=2 THEN GOSUB 920:GOTO 1220
 80 IF DR=1  THEN GOTO 1240
 90 GOSUB 920
100 GOTO 30
```

```
120 FOR Z=1 TO 4
130 X(Z)=0
140 Y(Z)=0
150 NEXT Z
160 FOR L=1 TO 3
170 S=0
180 T=0
190 FOR K=1 TO 3
200 IF A(L,K)=1 THEN S=S+1
210 IF B(L,K)=1 THEN T=T+1
220 NEXT K
230 IF S=0 THEN Y(T+1)=Y(T+1)+1
240 IF T=0 THEN X(S+1)=X(S+1)+1
250 NEXT L
260 FOR L=1 TO 3
270 T=0
280 S=0
290 FOR K=1 TO 3
300 IF A(K,L)=1 THEN S=S+1
310 IF B(K,L)=1 THEN T=T+1
320 NEXT K
330 IF S=0 THEN Y(T+1)=Y(T+1)+1
340 IF T=0 THEN X(S+1)=X(S+1)+1
350 NEXT L
360 GOSUB 430
370 GOSUB 530
380 IF X(4)=1 THEN FIN=1:RETURN
390 IF Y(4)=1 THEN FIN=2
400 E=128*Y(4)-63*X(3)+31*Y3)-15*X(2)+7*Y(2)
410 RETURN
430 T=0
440 S=0
450 FOR K=1 TO 3
460 T=T+A(K,K)
470 S=S+B(K,K)
480 NEXT K
490 IF S=0 THEN X(T+1)=X(T+1)+1
500 IF S=0 THEN Y(S+1)=Y(S+1)+1
510 RETURN
530 T=0
540 S=0
550 FOR K=1 TO 3
560 T=T+A(4-K,K)
570 S=S+B(4-K,K)
580 NEXT K
590 IF S=0 THEN X(T+1)=X(T+1)+1
600 IF T=0 THEN Y(S+1)=Y(S+1)+1
610 RETURN
```

```
630 M=-256
640 DR=1
650 FOR J=1 TO 3
660 FOR I=1 TO 3
670 IF A(I,J)=1 OR B(I,J)=1 THEN GOTO 740
680 DR=0
690 B(I,J)=1
700 GOSUB 120
710 IF FIN=1 THEN RETURN
720 IF E>M THEN M=E:A=I:B=J
730 B(I,J)=0
740 NEXT I
750 NEXT J
760 B(A,B)=1
770 RETURN

780 REM GET INPUT
790 INPUT"Your move (row col) ";A$
800 IF LEN(A$)<>2 THEN PING:GOTO 780
810 J=VAL(MID$(A$,1,1)):I=VAL(MID$(A$,2,1))
820 IF I<1 OR I>3 THEN PING:GOTO 780
830 IF J<1 OR J>3 THEN PING:GOTO 780
840 IF A(I,J)=1 THEN GOTO 890
850 IF B(I,J)=1 THEN GOTO 890
860 A(I,J)=1
870 REM INPUT OK
880 RETURN
890 PRINT"Position already occupied"
900 PING
910 GOTO 780

920 FOR J=1 TO 3
930 FOR I=1 TO 3
940 IF A(I,J)=1 THEN CURSET I*18+80,J*24,3:
    CHAR ASC("X"),0,1
950 IF B(I,J)=1 THEN CURSET I*18+80,J*24,3:
    CHAR ASC("O"),0,1
960 IF A(I,J)+B(I,J)=0 THEN CURSET I*18+80,J*24,3
    CHAR ASC(" "),0,1
970 NEXT I
980 NEXT J
990 RETURN
```

```
1020 DIM A(3,3)
1030 DIM B(3,3)
1040 DIM X(4):DIM Y(4)
1050 INK 7
1060 PAPER 0
1070 HIRES
1080 CURSET 109,10,2:DRAW 0,80,2
1090 CURSET 127,10,2:DRAW  0,80,2
1100 CURSET 90,40,2:DRAW 55,0,2
1110 CURSET 90,64,2:DRAW 55,0,2
1170 FIN=0
1180 DR=0
1190 RETURN


1220 PRINT "I WIN!!"
1230 GOTO 1270
1240 PRINT "DRAW!!"
1250 GOTO 1270
1260 PRINT "YOU WIN!!"
1270 INPUT "Another game Y/N";A$
1280 IF A$="Y" THEN RUN
1290 TEXT
```

# 19
# Fruit Machine


Here's a way of playing the fruit machine without spending a penny – your Oric gives you 100 pence to start with, takes 10 pence for every go, and awards you a sum between 5 pence and 50 pence every time you come up with a winning combination. You can give up while you are winning or carry on playing until you are broke.

Although short to type in, this program includes some really clever graphics techniques so that you see the drum of the fruit machine rotate smoothly, using only BASIC. In addition, there are sound effects that signal winning combinations. So listen out for the jackpot!

### How to play

There are four symbols in the display – cherries, banana, apple and bell. All the winning combinations are displayed on the screen while

you play. These combinations are winners wherever they occur on the line and not just as in the pattern suggested by the screen display but notice that where blanks occur you need some symbol *other* then the same type. To play just RUN and then answer "Y" every time you want another spin. If you do no answer "Y" then the computer will tell you how much money you are taking home with you. Once you run out of money the game is over.

### Subroutine structure

| | |
|---|---|
| 20 | Initialisation |
| 50 | Main play loop |
| 180 | Sets starting points of drum |
| 220 | Spins drum |
| 410 | Pay out routine |
| 470 | Displays title frame |
| 630 | Defines graphics characters in array C |
| 740 | Jackpot routine |
| 800 | Signals when broke |
| 1000 | Sets up graphics characters |
| 2000 | Win tone |

### Programming details

This program uses some very tricky programming techniques – which is why it achieves its effect in so short a length of BASIC. The patterns for each of the shapes are stored in an array, one line of dots to each array element. Each time the fruit machines drum is printed a different section of the array is used to *load* the user defined graphics. You can think of the section of the array that is used as being defined by a *window* which moves down by one row of dots each time the characters are printed. This produces the visual illusion of a smoothly rotating drum.

### Scope for improvement

If you like adding graphics and sound effects to programs there is scope in this game. For example, you could include a surround that looks like a one-armed-bandit and sounds of cascading coins and the drum rotating.

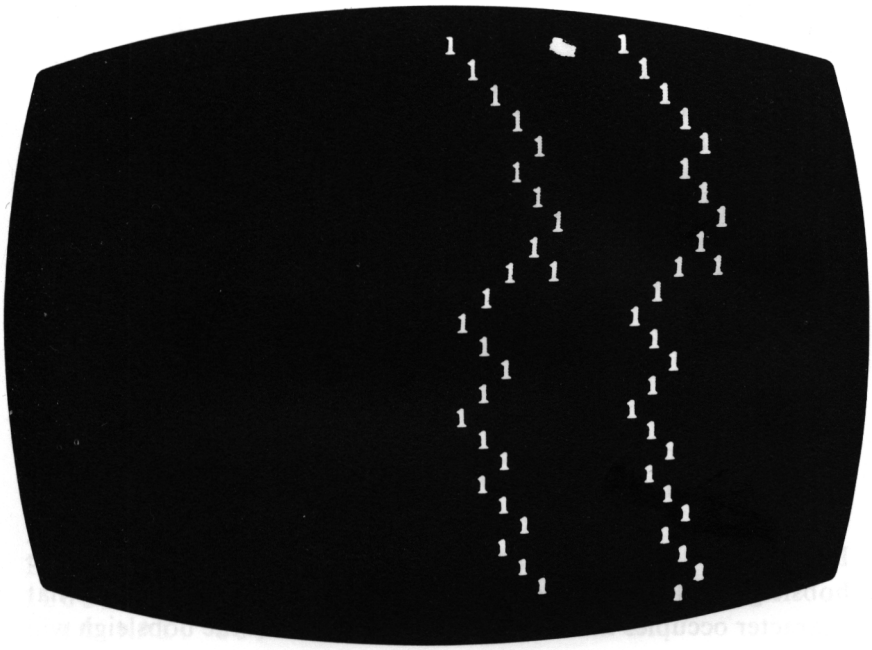
**Program**

```
 10 REM Fruit Machine

 20 GOSUB 630
 30 GOSUB 470
 40 M=100

 50 GOSUB 180
 60 M=M-10
 70 GOSUB 220
 80 GOSUB 410
 90 IF M<=0 THEN GOTO 800
100 MES$="You have "+STR$(M)+" p  "
110 PLOT 1,18,MES$
115 PING
120 PLOT 1,19,"Another spin Y/N":GET A$
130 IF A$="" THEN GOTO 120
140 IF A$="Y" THEN GOTO 50
150 MES$="You take home "+STR$(M)+" p":
    PLOT 1,20,MES$
160 WAIT 100
170 PRINT CHR$(17):STOP

180 X=INT(RND(1)*4)*10+1
190 Y=INT(RND(1)*4)*10+1
200 Z=INT(RND(1)*4)*10+1
210 RETURN

220 REM SPIN
230 S=INT(RND(1)*2)+1
240 FOR I=0 TO S*10
250 FOR Q=0 TO 7
260 POKE CH+ASC("$")*8+Q,C(Q+X)
270 POKE CH+ASC("%")*8+Q,C(Q+Y)
280 POKE CH+ASC("&")*8+Q,C(Q+Z)
290 NEXT Q
300 PLOT 7,16,"$ % &"
310 IF X=40 THEN X=0
320 IF Y=40 THEN Y=0
330 IF Z=40 THEN Z=0
340 X=X+1:Y=Y+1:Z=Z+1
350 NEXT I
360 X=X-1:Y=Y-1:Z=Z-1
370 RETURN
```

```
410 I=5
420 IF X=1 AND Y=1 AND Z=1 THEN GOSUB 740:
    RETURN
430 IF (X=11)+(Y=11)+(Z=11)<=-2 THEN M=M+25:
    GOSUB 2000
440 IF (X=21) +(Y=21)+(Z=21)<=-2 THEN M=M+10:
    GOSUB 2000
450 IF (X=1)+(Y=1)+(Z=1)<=-1 THEN M=M+5:
    GOSUB 2000
460 RETURN


470 CH=46080
530 GOSUB 1000
540 CLS:PRINT CHR$(17)
550 PLOT 5,1,"F R U I T"
560 PLOT 4,2,"M A C H I N E"
570 PLOT 3,4,"You have 1.00 to gamble"
580 PLOT 3,6,"Each spin costs  .10"
590 PLOT 5,9,"! ! ! wins 50p"
600 PLOT 5,10,"@ @ - wins 25p"
610 PLOT 5,11,"= [ [ wins 10p"
620 PLOT 5,12,"! -  - wins 5p"
625 RETURN

630 DATA 6,10,20,36,52,63,31,6,0,0
640 DATA 2,12,28,56,56,28,12,2,0,0
650 DATA 12,30,30,30,30,63,63,12,0,0
660 DATA 6,12,29,63,63,63,31,12,0,0
670 DATA 6,10,20,36,52,63,31,6,0,0
680 DIM C(48)
690 FOR I=1 TO 48
700 READ C:C(I)=C
710 NEXT I
720 RETURN

740 REM JACK
750 FOR I=1 TO 12
760 GOSUB 2000
770 NEXT I
780 M=M+50
790 RETURN
```

```
 800 REM BUST
 810 PLOT 1,20,"You are BUST"
 820 WAIT 500
 830 PRINT CHR$(17)
 840 STOP

1000 FOR Q=0 TO 7
1010 POKE CH+ASC("$")*8+Q,C(Q+1)
1020 POKE CH+ASC("!")*8+Q,C(Q+1)
1030 POKE CH+ASC("%")*8+Q,C(Q+10)
1040 POKE CH+ASC("@")*8+Q,C(Q+10)
1050 POKE CH+ASC("&")*8+Q,C(Q+20)
1060 POKE CH+ASC("[")*8+Q,C(Q+20)
1070 NEXT Q
1090 RETURN

2000 MUSIC 1,5,I,0
2010 PLAY 1,0,6,50
2020 WAIT 50
2030 PLAY 0,0,0,0
2040 WAIT 10
2050 RETURN
```

# 20
# Bobsleigh



In this game you have to steer your blue bobsleigh down a random course that winds its way downwards. You can choose whether to try a course that is easy to manoeuvre or one that is difficult – there are actually five levels of difficulty which govern the width of the course. If you crash you'll hear a dismal tone and that round of the game is over. Play this game to see how adept you are at keeping on course.

## How to play

The bobsleigh starts off at the top of the course and the course automatically moves past it. You have to steer the bobsleigh using the right and left arrow keys to ensure that you do not crash into the edges of the course. At the beginning of the game you have to select the difficulty level for the game. This governs the width of the course

with 1 producing the widest, and therefore easiest, and 5 the narrowest, and most difficult.

### Subroutine structure

| | |
|---|---|
| 20 | Start of game |
| 500 | Defines graphics characters and sets up colours |
| 1040 | Prints first part of track |
| 1220 | Main play loop |
| 1310 | Scrolls last part of track off screen |
| 5000 | Moves bobsleigh |
| 7000 | Title frame |
| 8000 | Win/lose messages and end of game |

### Programming details

The impression of the bobsleigh moving down the course is actually achieved by the course *scrolling* up the screen past the bobsleigh which only moves to left and right and is at a fixed vertical position. In line 5040 the SCRN function is used to test whether or not the bobsleigh has hit the side wall. This is done by detecting what character occupies the next printing position that the bobsleigh will move into.

Notice that while the track and bobsleigh are produced using PLOT which doesn't move the text cursor, each PLOT is accompanied by a PRINT, which does, and so causes the screen to scroll.

### Scope for improvement

You might like to add a tune-playing routine to this program like the one given in 'Oric Ledger'.

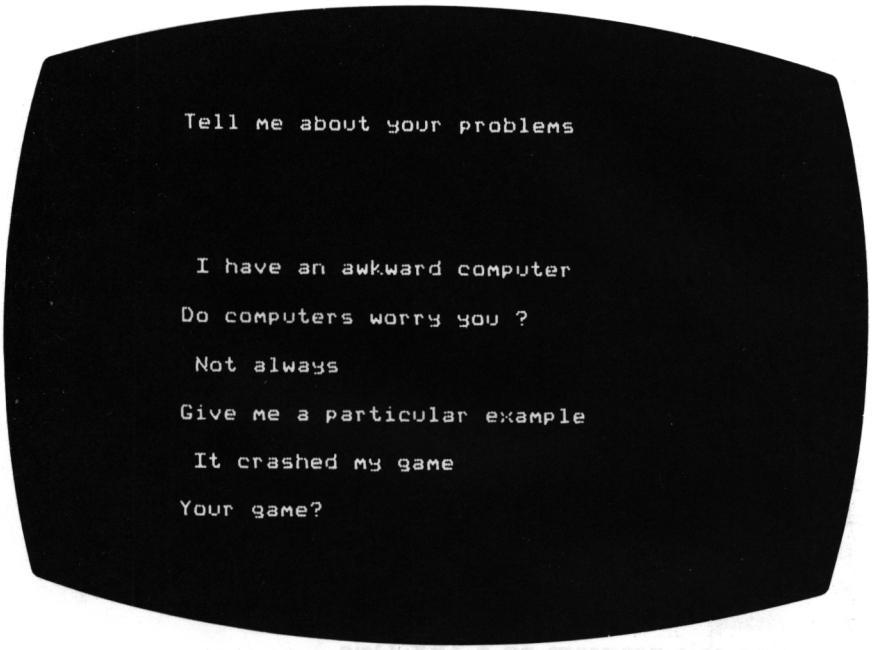
### Program

```
10 REM Bobsleigh

20 GOSUB 7000
```

```
 500 CHBAS=46080
 510 POKE CHBAS+(ASC("@")*8+0),16
 520 POKE CHBAS+(ASC("@")*8+1),20
 530 POKE CHBAS+(ASC("@")*8+2),52
 540 POKE CHBAS+(ASC("@")*8+3),62
 550 POKE CHBAS+(ASC("@")*8+4),63
 560 POKE CHBAS+(ASC("@")*8+5),31
 570 POKE CHBAS+(ASC("@")*8+6),15
 580 POKE CHBAS+(ASC("@")*8+7),7
 800 YB=1
1000 PAPER  7
1010 INK 0
1020 CLS
1030 PRINT CHR$(17);

1040 X=INT(RND(1)*10)+15
1050 XB=X+2
1100 FOR Y=1 TO 26
1110 PRINT:PLOT X,Y,"1"
1120 PLOT  X+D,Y,"1"
1130 X=X+(SGN(RND(1)-.5))
1140 IF X>30 THEN X=30
1150 IF X<1 THEN X=1
1160 NEXT Y
1200 PLOT XB,YB,"@"
1210 WAIT 100

1220 FOR Z=1 TO 100
1230 X=X+SGN(RND(1)-.5)
1240 IF X>31 THEN X=31
1250 IF X<1  THEN X=1
1260 PLOT X,26,"1"
1270 PLOT X+D,26,"1"
1280 PRINT
1290 GOSUB 5000
1300 NEXT Z
```

```
1310 FOR Z=1 TO 26
1320 PLOT 1,26,""
1330 PRINT
1340 GOSUB 5000
1350 NEXT Z
1360 GOSUB 8000

5000 A$=KEY$
5005 IF A$="" THEN A=0 ELSE A=ASC(A$)
5010 PLOT XB,YB-1," "
5020 IF A=8  THEN  XB=XB-1
5030 IF  A=9 THEN  XB=XB+1
5040 IF SCRN(XB,YB)=49 THEN EXPLODE:GOTO 8500
5050 PLOT XB,YB,"@@"
5060 RETURN


7000 INK 0
7010 PAPER 7
7020 CLS:PRINT
7030 PRINT SPC(10);"B O B  S L E I G H"
7035 FOR I=1  TO 5:PRINT:NEXT I
7040 PRINT SPC(4);"You must steer your bobsleigh"
7050 PRINT SPC(4);"down a dangerous course"
7060 PRINT SPC(4);"Select the danger level"
7065 FOR I=1 TO 10:PRINT:NEXT I
7070 INPUT "FROM 1 (SAFE) TO 5 (LETHAL) ";D
7080 IF D<1 OR D>5 THEN GOTO 7070
7090 D=9-D
7100 RETURN

8000 WAIT  300:CLS
8010 PRINT "CONGRATULATIONS,  YOU MADE IT"
8100 GOTO 9000
8500 WAIT 300:CLS
8510 PRINT "YOU CRASHED"
9000 PRINT CHR$(17)
9010 INPUT "ANOTHER RUN Y/N";A$
9020 IF A$="Y" THEN RUN
9030 INK 0
9040 PAPER 7
9050 CLS
9060 END
```

# 21
# Oric Oracle




Do you ever find yourself *talking* to your Oric? Well, if you do you may be disappointed that it never answers back. This program, however, changes all that and gives your Oric the chance to start a conversation with you. Although it may not be able to rival the agony aunts of the glossy magazines, your Oric is anxious to hear about your problems – and has some comments to offer.

Coping with the syntax of the English language is one of the very complicated problems with which this program has to contend. Programs like this one have been developed in order to extend our knowledge of how language works and how humans identify the key components of conversations. While these serious purposes are usually the province of *artificial intelligence* it is possible to have a good deal of fun trying to conduct a dialogue with your Oric.

### How to use this program

The computer opens each conversation in the same way – by inviting you to tell it your problems. You can give any reply that you wish to and after a few moments delay your Oric will respond. Try to say more than just "Yes" or "No" when you make further responses but equally, don't say too much at a time. If you type about a lineful each time you ought to be able to keep a reasonable conversation going.

### Typing tips

As the computer has to match your sentences against its vocabulary it is also very important to be careful about your spelling. If you type in either the initial program or subsequent responses with misspellings the computer won't recognise your messages and you won't receive any sensible answers. The apostrophe is the only punctuation mark that should be used in dialogues with the Oric.

This is by far the longest program in this collection and it is an especially trying one to type in because of all its long lines and frequent repetition. If you want a short cut to using it remember that there are cassette tapes available.

### Running this program on a 16K Oric

This is the only problem in this book that does not fit comfortably into a 16K Oric. But don't panic – there is an easy way around the problem. When you get an "OUT OF MEMORY ERROR" while entering the program, type "GRAB" in direct mode (that is, without a line number) then add

```
5 GRAB
```

to the program. The GRAB command allows the program to use the memory normally set aside for the high resolution graphics screen.

### Subroutine structure

| | |
|---|---|
| 15 | Main program loop |
| 100 | Initial message and set up |
| 400 | Input human's sentence |

560 Divides sentences into words
650 Changes tense/pronouns
820 Tense/pronouns data
920 Finds keywords in sentence
1070 Keywords data
1250 Keyword responses
2210 Prints computer's response
2270 Requests sensible input

## Programming details

This program works by taking a sentence and splitting it down into individual words and then responding according to a list of keywords that it searches for in each sentence. So if, for example, your sentence contains the word 'why', the response 'Some questions are difficult to answer' will always be given by the computer. When the computer fails to find a specific reply to a sentence one of a number of responses is selected at random.

Although this technique sounds simple, the actual details of the program are really quite tricky as, amongst other things, the computer has to deal with tense changes and with the syntax of pronouns. It is therefore quite a difficult program to write or to modify extensively. Equally, despite the apparent simplicity of its underlying technique, it succeeds in making plausible responses on a surprising number of occasions.

## Scope for improvement

If you wish to add to the list of keywords that the computer recognises, you need to notice how, in subroutine 1070, the keywords are paired with the line number of the subroutine that responds to them. Also it is important to be aware of the priorities assigned to each keyword. If two keywords are present in a sentence then the one first in the list will be acted upon.

## Program

```
10 REM Oric Oracle
```

```
 15 DIM W(20,2)
 20 GOSUB 100
 30 GOSUB 400
 40 GOSUB 560
 50 GOSUB 920
 60 IF NM<>0 THEN ON NM GOSUB 1250,1270,1290,
    1310,1730,1780,1690,1710,1610
 65 IF NM<>0 AND NM>9 THEN ON NM-9 GOSUB 1380,
    2140,1330,1470,1900,2040,1800
 66 IF NM<>0 AND NM>16 THEN ON NM-16 GOSUB
    1820,1840,1330,1880,1860
 70 GOSUB 2210
 80 GOTO 30

100 TEXT:CLS:PRINT CHR$(20)
110 PRINT:PRINT "Hi! My name is Oric"
120 PRINT
130 PRINT "I would like you to talk to me"
140 PRINT "but I don't have ears so will"
150 PRINT "you type sentences on my"
160 PRINT "keyboard"
170 PRINT
180 PRINT "Don't use any punctuation apart"
190 PRINT "from apostrophies which are"
200 PRINT "important"
210 PRINT
220 PRINT "When you have finished typing"
230 PRINT "press RETURN"
240 PRINT
250 PRINT "Tell me about your problems"
255 FOR I=1 TO 10:PRINT:NEXT I
260 R$=""
270 M$=""
280 D$=""
290 DIM N$(3)
300 N$(1)="Please go on"
310 N$(2)="I'm not sure I understand you"
320 N$(3)="Tell me more"
330 DIM I$(3)
340 I$(1)="Let's talk some more about your"
350 I$(2)="Earlier you spoke of your"
360 I$(3)="Does that have anything to do
    with your"
370 DIM J$(2)
380 J$(1)="Are you just being negative"
390 J$(2)="I see"
399 RETURN
```

```
400 A$=""
410 B$=KEY$
415 IF B$="" THEN GOTO 410
420 IF ASC(B$)=13 THEN PRINT:GOTO 480
430 IF ASC(B$)=127 AND A$<>"" THEN
    A$=LEFT$(A$,LEN(A$)-1):GOTO
440 IF ASC(B$)<32 OR ASC(B$)>126 THEN GOTO 400
450 A$=A$+B$
460 PLOT 2,25,A$+" "
470 GOTO 410
480 IF RIGHT$(A$,1)=" " THEN
    A$=LEFT$(A$,LEN(A$)-1):GOTO 480
481 FOR I=1 TO LEN(A$)
482 A=ASC(MID$(A$,I,1))
485 IF A<91 AND A>64 THEN A$=LEFT$(A$,I-1)+
    CHR$(A+32)+MID$(A$,I+1)
486 NEXT I
490 IF A$=R$ THEN PRINT "You're repeating
    yourself!":PRINT:GOTO 400
500 R$=A$
510 IF A$="" THEN GOTO 400
520 RETURN

560 REM FIND WORDS
570 N=1
580 B=0
590 FOR I=1 TO LEN(A$)
600 IF (MID$(A$,I,1)=" " OR MID$(A$,I,1)=",")
    AND B=0 THEN B=1
610 IF ((MID$(A$,I,1)<>" " AND MID$(A$,I,1)<>",")
    AND B<=1) THEN W(N,1)=I:B=2
620 IF (MID$(A$,I,1)=" " OR MID$(A$,I,1)=",")
    AND B=2 THEN W(N,2)=I-1:N=N+1
625 IF (MID$(A$,I,1)=" " OR MID$(A$,I,1)=",")
    AND B=2 THEN B=0
630 NEXT I
640 W(N,2)=LEN(A$)
```

```
 650 FOR I=1 TO N
 660 RESTORE
 670 READ B$
 680 IF B$="s" THEN GOTO 800
 690 IF B$<>MID$(A$,(W(I,1)),(W(I,2)-W(I,1)+1))
     THEN GOTO 780
 700 READ C$
 710 A$=LEFT$(A$,(W(I,1)-1))+C$+
     RIGHT$(A$,(LEN(A$)-W(I,2)))
 720 W(I,2)=W(I,2)+LEN(C$)-LEN(B$)
 730 FOR J=I+1 TO N
 740 W(J,2)=W(J,2)+LEN(C$)-LEN(B$)
 750 W(J,1)=W(J,1)+LEN(C$)-LEN(B$)
 760 NEXT
 770 GOTO 800
 780 READ B$
 790 GOTO 670
 800 NEXT I
 810 RETURN

 820 DATA "my","your*","i","you*"
 830 DATA "mum","mother","dad","father"
 840 DATA "dreams","dream","you","i*","me","you*"
 850 DATA "your","my*","myself","yourself*"
 860 DATA "yourself","myself*","i'm","you're*"
 870 DATA "you're","i'm","am","are*"
 880 DATA "i'm","you're"
 890 DATA "were","was"
 900 DATA "are","am"
 910 DATA "s","s"

 920 RESTORE:FOR I=1 TO 34:READ Q$:NEXT I
 930 READ B$,NM
 940 IF B$="s" THEN GOTO 1010
 950 I=1
 960 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))<>B$
     THEN GOTO 990
 970 T$=RIGHT$(A$,LEN(A$)-(W(I,2)))
 980 RETURN
 990 I=I+1:IF I<=N THEN GOTO 960
1000 GOTO 930
1010 NM=0
1020 IF M$<>"" THEN GOTO 1050
1030 P$=N$(INT(RND(1)*3)+1)
1040 RETURN
1050 P$=I$(INT(RND(1)*3)+1)+M$
1060 RETURN
```

```
1070 DATA "computer",1,"machine",1,"program",1
1080 DATA "like",2,"same",2,"alike",2
1090 DATA "if",3,"everybody",4
1100 DATA "can",5,"certainly",6
1110 DATA "how",7,"because",8
1120 DATA "always",9
1130 DATA "everyone",4,"nobody",4
1140 DATA "was",10
1150 DATA "i*",11
1160 DATA "no",12
1170 DATA "your*",13
1180 DATA "you're*",14,"you*",15
1190 DATA "hello",16,"maybe",17
1200 DATA "my*",18,"no",19
1210 DATA "yes",6,"why",20
1220 DATA "perhaps",17,"sorry",21
1230 DATA "what",20
1240 DATA "s",0
```

```
1250 P$="Do computers worry you ?"
1260 RETURN
1270 P$="In what way ?"
1280 RETURN
1290 P$="Why talk of possibilities"
1300 RETURN
1310 P$="Really "+B$+" ?"
1320 RETURN
1330 IF I=N THEN GOTO 1360
1340 I=I+1
1350 IF MID$(A$,W(I,1),(W(I,2)-W(I,2)-W(I,1)+1))
     ="one" THEN B$=B$+"one"
1355 IF MID$(A$,W(I,1),(W(I,2)-W(I,2)-W(I,1)+1)
     ="one" THEN GOTO 1310
1360 P$=J$(INT(RND(1)*2)+1)
1370 RETURN
1380 IF I=N THEN GOTO 2270
1390 I=I+1
1400 IF I>N THEN GOTO 1020
1410 IF MID$(A$,W(I,1),W(I,2)-W(I,1)+1)<>"you*"
     THEN GOTO 1440
1420 P$="What if you were "+RIGHT$(A$,LEN(A$)
     -W(I,2))+" ?"
1430 RETURN
1440 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))<>"i*"
     THEN GOTO 1020
1450 P$="Would you like to believe I was "
     +RIGHT$(A$,(LEN(A$)-(W(I,1)+1)))
1460 RETURN
1470 I=I+1
1480 IF I>N THEN GOTO 1360
1490 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="mother"
     THEN GOTO 1590
1500 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="father"
     THEN GOTO 1590
1510 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="sister"
     THEN GOTO 1590
1520 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="brother"
     THEN GOTO 1590
1530 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="wife"
     THEN GOTO 1590
1540 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="husband"
     THEN GOTO 1590
1550 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))=
     "children" THEN GOTO 1590
```

```
1560 IF LEN(T$)>10 THEN M$=T$+" ?"
1570 P$="your "+T$+" ?"
1580 RETURN
1590 P$="Tell me more about your family"
1600 RETURN
1610 P$="Give me a particular example"
1620 RETURN
1630 I=I+1
1640 IF I>N THEN P$="Am I what ?":RETURN
1650 P$="Why are you interested in whether I am "
   +RIGHT$(A$,W(I,1)+1)
1655 P$=P$+" or not?"
1660 RETURN
1670 P$="Do you think you are "+RIGHT$(A$,LEN(A$)
     -W(I,2))
1680 RETURN
1690 P$="Why do you ask ?"
1700 RETURN
1710 P$="Tell me any other reasons"
1720 RETURN
1730 I=I+1
1740 IF I>N THEN P$="what ?":RETURN
1750 IF MID$(A$,W(I,1),W(I,2)-W(I,1)+1)<>"i*"
     THEN GOTO 1760
1755 P$="Do you believe I can "+RIGHT$(A(LEN(A$)
     -W(I,2)))+" ?":RETURN
1760 IF MID$(A$,W(I,1),(W(I,2)-W(I,1))+1)<>"you*"
     THEN GOTO 1010
1770 P$="Do you believe you can "+RIGHT$(A$,
     LEN(A$)-W(I,2))+" ?":RETURN
1780 P$="You seem very positive"
1790 RETURN
1800 P$="Pleased to meet you - let's talk about
     your problems"
1810 RETURN
1820 P$="Could you try to be more positive"
1830 RETURN
1840 P$="Why are you concerned about my "+T$
1850 RETURN
1860 P$="You don't have to apologise to me"
1870 RETURN
1880 P$="Some questions are difficult to answer..."
1890 RETURN
1900 I=I+1
1910 IF I>N THEN GOTO 1020
```

```
1920 P$="I am sorry to hear that you are "
     +MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))
1930 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="sad"
     THEN RETURN
1940 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="unhappy"
     THEN RETURN
1950 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))
     ="depressed" THEN RETURN
1960 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="sick"
     THEN RETURN
1970 P$="How have I helped you to be "
     +MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))
1980 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="happy"
     THEN RETURN
1990 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="elated"
     THEN RETURN
2000 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="glad"
     THEN RETURN
2010 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="better"
     THEN RETURN
2020 P$="Is it because you are "+MID$(A$,W(I,1),
     W(I,2)-W(I,1)+1)
2025 P$=P$+" you would like to talk to me ?"
2030 RETURN
2040 IF I=1 THEN GOTO 2060
2050 IF MID$(A$,W(I-1,1),(W(I-1,2)-W(I-1,I)+1))
     ="are*" THEN GOTO 1670
2060 I=I+1
2070 IF I>N THEN GOTO 2270
2080 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="are*"
     THEN GOTO 1900
2090 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="want"
     THEN GOTO 2096
2095 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))<>"need"
     THEN GOTO 2100
2096 P$="What would it mean if you got "
     +RIGHT$(A$,LEN(A$)-W(I,2)):RETURN
2100 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))<>"think"
     THEN GOTO 2110
2105 P$="Do you really think so":RETURN
2110 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="can't"
     THEN GOTO 2116
2115 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))<>"cannot"
     THEN GOTO 2120
2116 P$="How do you know you can't"
     +RIGHT$(A$,LEN(A$)-W(I,2)):RETURN
```

```
2120 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))<>"feel"
     THEN GOTO 1020
2130 P$="Tell me more about how you feel":RETURN
2140 IF I-1<1 THEN GOTO 2160
2145 I=I-1
2150 IF MID$(A$,W(I,1),(W(I,2)-W(I,1)+1))="am"
     THEN GOTO 1630
2155 I=I+1
2160 I=I+1
2170 IF I>N THEN P$="What am I ?":RETURN
2180 IF MID$(A$,W(I,1),W(I,2)-W(I,1)+1)<>"am"
     THEN GOTO 2190
2185 P$="Why do you think so ?":RETURN
2190 P$="Is that what you think of me ?":RETURN
2200 RETURN

2210 FOR J=1 TO LEN(P$)
2220 IF MID$(P$,J,1)="*" THEN GOTO 2240
2230 PRINT MID$(P$,J,1);
2240 NEXT J
2250 PRINT:PRINT:PRINT
2260 RETURN

2270 P$="PLEASE TALK SENSIBLY !"
2280 RETURN
```

**THE ORIC-1 PROGRAMMER**
M. James and S. M. Gee
0 246 12157 2

**ORIC MACHINE CODE HANDBOOK**
Paul Kaufman
0 246 12150 5

## The TI 99/4A

**GET MORE FROM THE TI99/4A**
Garry Marshall
0 246 12281 1

## The VIC 20

**GET MORE FROM THE VIC 20**
Owen Bishop
0 246 12148 3

**THE VIC 20 GAMES BOOK**
Owen Bishop
0 246 12187 4

## The ZX Spectrum

**THE ZX SPECTRUM And How To Get The Most From It**
Ian Sinclair
0 246 12018 5

**THE SPECTRUM PROGRAMMER**
S. M. Gee
0 246 12025 8

**THE SPECTRUM BOOK OF GAMES**
M. James, S. M. Gee and K. Ewbank
0 246 12047 9

**INTRODUCING SPECTRUM MACHINE CODE**
Ian Sinclair
0 246 12082 7

**SPECTRUM GRAPHICS AND SOUND**
Steve Money
0 246 12192 0

**THE ZX SPECTRUM How to Use and Program**
Ian Sinclair
0 586 06104 5

**Learning is Fun! 40 EDUCATIONAL GAMES FOR THE SPECTRUM**
Vince Apps
0 246 12233 1

## The ZX81

**THE ZX81 How to Use and Program**
S. M. Gee and Mike James
0 586 06105 3

## Which Computer?

**CHOOSING A MICROCOMPUTER**
Francis Samish
0 246 12029 0

## Languages

**COMPUTER LANGUAGES AND THEIR USES**
Garry Marshall
0 246 12022 3

**EXPLORING FORTH**
Owen Bishop
0 246 12188 2

## Machine Code

**Z-80 MACHINE CODE FOR HUMANS**
Alan Tootill and David Barrow
0 246 12031 2

**6502 MACHINE CODE FOR HUMANS**
Alan Tootill and David Barrow
0 246 12076 2

## Using Your Micro

**COMPUTING FOR THE HOBBYIST AND SMALL BUSINESS**
A. P. Stephenson
0 246 12023 1

**DATABASES FOR FUN AND PROFIT**
Nigel Freestone
0 246 12032 0

**SIMPLE INTERFACING PROJECTS**
Owen Bishop
0 246 12026 6

**INSIDE YOUR COMPUTER**
Ian Sinclair
0 246 12235 8

## Programming

**THE COMPLETE PROGRAMMER**
Mike James
0 246 12015 0

**PROGRAMMING WITH GRAPHICS**
Garry Marshall
0 246 12021 5

## Word Processing

**CHOOSING A WORD PROCESSOR**
Francis Samish
0 246 12347 8

**WORD PROCESSING FOR BEGINNERS**
Susan Curran
0 246 12353 2

## 21 QUALITY GAMES FOR YOUR ORIC!

**Here is a compendium of twenty-one exciting games with something for all the family. The selection comprises a combination of old favourites such as Alien Invaders, Squash and Bobsleigh, and exciting new ideas including Commando Jump, Across the Ravine and Sheepdog Trials. There are also computer implementations of the traditional pastimes, Noughts and Crosses and Dice, a challenging board game, Capture the Quark, and a conversational game in which you can talk to your Oric. Full use is made of the Oric's colour graphics, its sound and its special facilities. All the programs are fully tested and crash-proofed, and listed straight from working versions.**

**The special feature of this book is that it aims to develop your own programming techniques at the same time as providing you with a selection of high quality BASIC games. To this end each program is well documented with a clear outline of its subroutine structure and an explanation of the special techniques employed. This book not only shows you how to play the games and enjoy them, it enables you to improve your BASIC programming by letting you in on 'trade secrets' professional programmers use.**

*The Authors*

These three authors have collaborated on many programming projects, and between them have written several successful books. They are also regular contributors to *Electronics and Computing Monthly* and other magazines.



**GRANADA PUBLISHING**
Printed in Great Britain 0 246 12155 6

**£5.95 net**

JAMES, GEE AND EWBANK **THE ORIC BOOK OF GAMES** GRANADA